CROSSTALK

November 2008 The Journal of Defense Software Engineering Vol. 21 No. 11

INTEROPERABILITY

Interoperability

Software Engineering Technology

Editor's Note: In this issue, the concepts of System of Systems and Systems of Systems are used. Two articles (Anderson and Boxer's as well as Colombi, Cohee, and Turner's) utilize the acronym SoS for System of Systems and a third (Dahmann, Rebovich Jr., and Lane) uses the acronym SoS for Systems of Systems. While CrossTalk tries to maintain consistency, we have made an exception in this issue based on the authors' disparate requests.

On The Cover

Cover Design by Kent Bingham

Additional art services provided by Janna Jensen

Departments

CrossTalk

Co-Sponsors:
DoD-CIO *The Honorable John Grimes*
OSD (AT&L) *Kristen Baldwin*
NAVAIR *Jeff Schwalb*
76 SMXG *Daniel Goddard*
309 SMXG *Karl Rogers*
DHS *Joe Jarzombek*

Staff:
Managing Director *Brent Baxter*
Publisher *Kasey Thompson*
Managing Editor *Drew Brown*
Associate Editor *Chelene Fortier-Lozancich*
Publishing Coordinator *Nicole Kentta*
Phone (801) 775-5555
E-mail stsc.customerservice@hill.af.mil
CrossTalk Online www.stsc.hill.af.mil/crosstalk

CrossTalk, The Journal of Defense Software Engineering is co-sponsored by the Department of Defense Chief Information Office (DoD-CIO); the Office of the Secretary of Defense (OSD) Acquisition, Technology and Logistics (AT&L); U.S. Navy (USN); U.S. Air Force (USAF); and the U.S. Department of Homeland Security (DHS). DoD-CIO co-sponsor: Assistant Secretary of Defense (Networks and Information Integration). OSD (AT&L) co-sponsor: Software Engineering and System Assurance. USN co-sponsor: Naval Air Systems Command. USAF co-sponsors: Oklahoma City-Air Logistics Center (ALC) 76 Software Maintenance Group (SMXG); and Ogden-ALC 309 SMXG. DHS co-sponsor: National Cyber Security Division in the National Protection and Programs Directorate.

The USAF Software Technology Support Center (STSC) is the publisher of CrossTalk, providing both editorial oversight and technical review of the journal. CrossTalk's mission is to encourage the engineering development of software to improve the reliability, sustainability, and responsiveness of our warfighting capability.

S T S C

Subscriptions: Send correspondence concerning subscriptions and changes of address to the following address. You may e-mail us or use the form on p. 17.

517 SMXS/MXDEA
6022 Fir AVE
BLDG 1238
Hill AFB, UT 84056-5820

Article Submissions: We welcome articles of interest to the defense software community. Articles must be approved by the CrossTalk editorial board prior to publication. Please follow the Author Guidelines, available at <www.stsc.hill.af.mil/crosstalk/xtlkguid.pdf>. CrossTalk does not pay for submissions. Published articles remain the property of the authors and may be submitted to other publications. Security agency releases, clearances, and public affairs office approvals are the sole responsibility of the author and their organizations.

Reprints: Permission to reprint or post articles must be requested from the author or the copyright holder and coordinated with CrossTalk.

Trademarks and Endorsements: This Department of Defense (DoD) journal is an authorized publication for members of the DoD. Contents of CrossTalk are not necessarily the official views of, or endorsed by, the U.S. government, the DoD, the co-sponsors, or the STSC. All product names referenced in this issue are trademarks of their companies.

CrossTalk Online Services: See <www.stsc.hill.af.mil/crosstalk>, call (801) 777-0857 or e-mail <stsc.webmaster@hill.af.mil>.

Back Issues Available: Please phone or e-mail us to see if back issues are available free of charge.

Integrating Software and Systems Engineering to Promote Interoperability

As the Acting Director of Software and Systems Engineering in the Office of the Deputy Under Secretary of Defense (Acquisition and Technology), I occupy a uniquely advantageous position to witness the challenges that interoperability imposes on the engineering community. In particular, I have observed that very distinct cultures distinguish the software engineering and systems engineering communities and the subtle differences in their perspectives, and how these have undermined the DoD's ability to develop solutions to warfighter needs. Although systems engineering, as a discipline, nominally encompasses software, its heritage is hardware-oriented and favors a product-oriented perspective and functional decomposition. By comparison, the software community—with roots in embedded systems, information technology, and command and control domains—embraces layered architectures and process-focused development perspectives.

Perpetuation of functionally stove-piped policies and organizational structures has permitted each community's worldview to somewhat peacefully co-exist by limiting the amount of interaction. Today's systems cannot ignore the need to interoperate. Information technology has allowed us to shift the balance of control from hardware to software and enabled an ad-hoc composition of systems which are not specifically designed to interoperate. We face warfighter demand for capabilities that are joint and adaptable. This requires acknowledgement of an enhanced systems engineering imperative to seamlessly integrate hardware, software, and human factors and enable system of system solutions. We must synthesize the most thoughtful perspectives of software and systems engineering to capitalize on technology and deliver integrated capabilities to our customers.

This issue of CROSSTALK addresses some of these compelling challenges as we strive to improve integration/interoperability. In *Systems Engineering for Capabilities*, Dr. Judith S. Dahmann, George Rebovich Jr., and Jo Ann Lane adapt systems engineering concepts for the development of capabilities. Dr. John Colombi, Maj. Brannen C. Cohee, and Maj. Chuck W. Turner discuss—in *Interoperability Test and Evaluation: A System of Systems Field Study*—the policy and practice of testing for interoperability in a system of systems context. Shamlan Siddiqi's *Key Transformational Techniques to Achieve Enterprise-Scale Interoperability* details the use of service-oriented architecture design principles and an agile methodology. William B. Anderson and Philip Boxer's *Modeling and Analysis of Interoperability Risks in Systems of Systems Environments* describes how their techniques in an interoperability risk probe found gaps in the ability of a modernization program to react to changing demands. In *Quality and Cost – It's Not Either/Or: Making the Case With Cost of Quality*, George Webb and LTC Nanette Patton describe the application of "cost of quality" principles to permit the balancing of these attributes.

I cannot overemphasize the importance and criticality of the need for the software and systems engineering communities to jointly confront the challenges of fielding systems that are affordable, sustainable, and interoperable.

Kristin Baldwin
Acting Director, Software and Systems Engineering
Office of the Deputy Under Secretary of Defense (Acquisition and Technology)

Systems Engineering for Capabilities

Dr. Judith S. Dahmann and George Rebovich Jr.
The MITRE Corporation

Jo Ann Lane
University of Southern California

With the increased emphasis on capabilities and networking, the DoD is recognizing the criticality of effective end-to-end performance of systems of systems (SoS) to meet user needs. While acquisition continues to focus on systems, systems requirements are increasingly based on assessment of gaps in user capabilities and in priority areas; there is an increasing focus on integration across systems to enable capabilities. Thus, the role of systems engineering (SE) is expanding to the engineering of SoS that provide user capabilities. This article discusses the shape of SoS in the DoD today. It outlines a recent initiative to provide guidance on the application of SE processes to the definition and evolution of SoS.

Beginning with the 2000 Quadrennial Defense Review (QDR) [1], the DoD has been reorienting force development processes to the identification and support of user capabilities, with an emphasis on agile composition of systems to meet a range of changing user needs. The Joint Capabilities Integration and Development System [2] was created in 2003 to identify capability needs in terms of functional concepts and validated material needs in terms of capability gaps. In parallel, the DoD 5000 [3] recognized capability areas and spawned initial efforts to develop road maps for capabilities. The 2006 QDR [4] continued this trajectory and, based on institutional reform and governance recommendations, the DoD has created capability portfolio managers in a further effort to view investments within the broader context of user capabilities [5].

At the same time, the DoD recognized the importance of SE as a key enabler of systems acquisition. Policy emphasized the importance of SE and renewed emphasis on technical planning and authorities [6, 7, 8], bringing attention to the robust SE in systems acquisitions. More recently, the SE community has recognized the need for discipline and structure in the engineering of SoS.

The Shape of SoS in the DoD Today

An SoS is defined as a set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities [9].

While DoD acquisition largely continues to emphasize development of individual systems, it has been increasingly recognized that for priority capabilities it is important to provide management and SE to the ensembles of systems which work together to support user capability needs. From a networking perspective, a set of DoD policies have been promulgated with the objective of providing the requisite infrastructure to support communications and data exchange among systems [10, 11].

In the DoD today, there are several types of SoS [12, 13], as shown in Table 1. The U.S. Army's Future Combat Systems is the best-known example of *directed SoS*. Communities of interest are good examples of DoD *collaborative SoS*, and the Global Information Grid is the predominant DoD *virtual SoS*.

Increasingly, the DoD is facing the challenges of *acknowledged SoS* by recognizing the need for capability management and SE at the SoS level while maintaining the management and technical autonomy of the systems contributing to the SoS capability objectives. Examples of this type of SoS are the Missile Defense Agency's Ballistic Missile Defense System, the Air Force's Air Operations Center, and the Naval Integrated Fire Control-Counter Air capability.

In the DoD, a typical strategy for providing end-to-end support for new capability needs is to add functionality to the inventory. In most cases, these systems continue to be needed for their original requirements. Consequently, the ownership or management of these systems remains unchanged and they continue to evolve based on their own development and requirements processes and independent funding.

The dual levels of management, objectives, and funding result in management challenges for both the SoS and the

Table 1: *Types of SoS*

Type	Definition
Virtual	Virtual SoS lack a central management authority and a centrally agreed-upon purpose for the system of systems. Large-scale behavior emerges—and may be desirable—but this type of SoS must rely upon relatively invisible mechanisms to maintain it.
Collaborative	In collaborative SoS, the component systems interact more or less voluntarily to fulfill agreed-upon central purposes. The Internet is a collaborative system. The Internet Engineering Task Force works out standards but has no power to enforce them. The central players collectively decide how to provide or deny service, thereby providing some means of enforcing and maintaining standards.
Acknowledged	Acknowledged SoS have recognized objectives, a designated manager, and resources for the SoS; however, the constituent systems retain their independent ownership, objectives, funding, as well as development and sustainment approaches. Changes in the systems are based on collaboration between the SoS and the system.
Directed	Directed SoS are those in which the integrated system of systems is built and managed to fulfill specific purposes. It is centrally managed during long-term operation to continue to fulfill those purposes as well as any new ones the system owners might wish to address. The component systems maintain an ability to operate independently, but their normal operational mode is subordinated to the central managed purpose.

Aspect of Environment	System	Acknowledged SoS
Management and Oversight		
Stakeholder Involvement	Clearer set of stakeholders.	Stakeholders at both system level and SoS levels (including the system owners, with competing interests and priorities); in some cases, the system stakeholder has no vested interest in the SoS; all stakeholders may not be recognized.
Governance	Aligned project manager and funding.	Added levels of complexity due to management and funding for both the SoS and individual systems; SoS does not have authority over all the systems.
Operational Environment		
Operational Focus	Designed and developed to meet operational objectives.	Called upon to meet a set of operational objectives using systems whose objectives may or may not align with the SoS objectives.
Implementation		
Acquisition	Aligned to acquisition categories milestones, documented requirements, SE with a SE plan.	Added complexity due to multiple system lifecycles across acquisition programs, involving legacy systems, developmental systems, new developments, and technology insertion; typically have stated capability objectives up front which may need to be translated into formal requirements.
Test and Evaluation	Test and evaluation of the system is generally possible.	Testing is more challenging due to the difficulty of synchronizing across multiple systems' life cycles, given the complexity of all the moving parts and potential for unintended consequences.
Engineering and Design Considerations		
Boundaries and Interfaces	Focuses on boundaries and interfaces for the single system.	Focus on identifying the systems that contribute to the SoS objectives and enabling the flow of data, control, and functionality across the SoS while balancing needs of the systems.
Performance and Behavior	Performance of the system to meet specified objectives.	Performance across the SoS that satisfies SoS user capability needs while balancing needs of the systems.

Table 2: *Comparison of Systems and SoS*

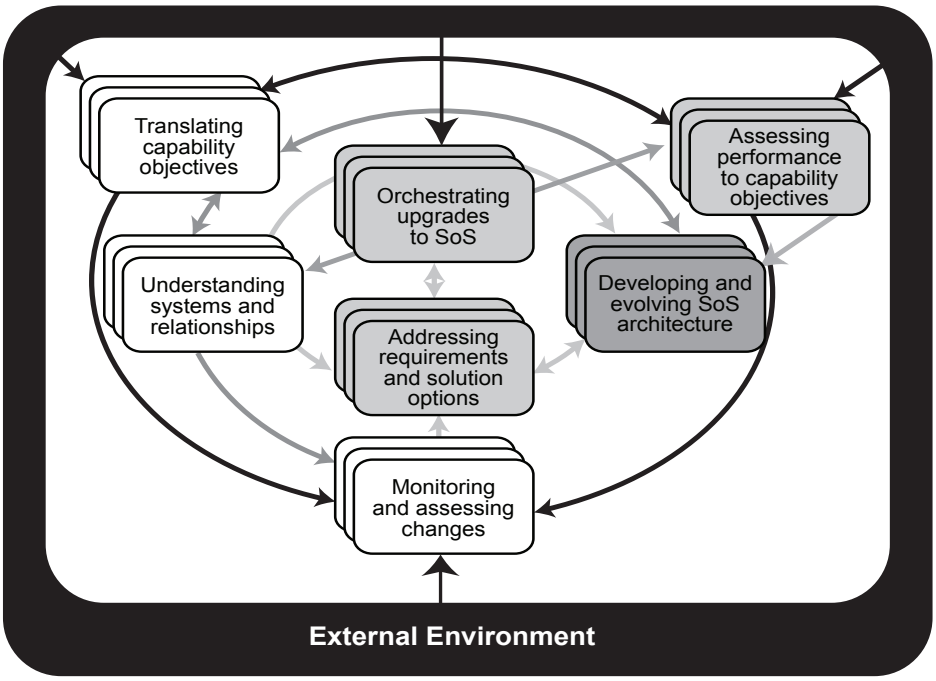
systems, especially when their objectives are not well-aligned. In turn, these management challenges pose technical challenges for the systems engineers, especially the SoS. Table 2 (from [14]) lists some additional observations regarding the differences between systems and SoS.

Core Elements of SE for SoS

To support SE for SoS, the Acquisition, Technology and Logistics (AT&L) Directorate of System and Software Engineering has developed the "Systems Engineering Guide for Systems of Systems" [14]. The guide is based on the experiences of SE practitioners and researchers currently working with SoS. It identifies seven core elements of SoS SE (as shown in Figure 1) along with their interrelationships.

Using these core elements as a frame of reference, [14] describes how the 16 SE processes from [9] (described in Table 3, next page) are applied in SoS. As systems engineers support SoS SE, they leverage these basic engineering processes. These processes, which were developed for the engineering of individual systems, essentially provide a set of tools for the systems engineers as they face the challenges of SoS engineering. The nature of the SoS environment affects the way these processes are employed to support SoS SE. The SE processes which apply to each of the core elements are

Figure 1: *SoS SE Core Elements and Their Relationships*



Technical Processes
Requirements Development takes all inputs from relevant stakeholders and translates the inputs into technical requirements.
Logical Analysis is the process of obtaining sets of logical solutions to improve understanding of the defined requirements and the relationships among the requirements (e.g., functional, behavioral, temporal).
Design Solution translates the outputs of the Requirements Development and Logical Analysis processes into alternative design solutions and selects a final design solution.
Implementation is the process that actually yields the lowest-level system elements in the system hierarchy. The system element is made, bought, or reused.
Integration is the process of incorporating the lower-level system elements into a higher-level system element in the physical architecture.
Verification confirms that the system element meets the design-to or build-to specifications. It answers the question "Did you build it right?"
Validation answers the question of "Did you build the right thing?"
Transition is the process applied to move … the end-item system to the user.
Technical Management Processes
Decision Analysis provides the basis for evaluating and selecting alternatives when decisions need to be made.
Technical Planning ensures that the SE processes are applied properly throughout a system's life cycle.
Technical Assessment measures technical progress and the effectiveness of plans and requirements.
Requirements Management provides traceability back to user-defined capabilities.
Risk Management ensures program cost, schedule, and performance objectives are achieved at every stage in the life cycle and communicated to all stakeholders the process for uncovering, determining the scope of, and managing program uncertainties.
Configuration Management is the application of sound business practices to establish and maintain consistency of a product's attributes with its requirements and product configuration information.
Data Management addresses the handling of information necessary for or associated with product development and sustainment.
Interface Management ensures interface definition and compliance among the elements that compose the system, as well as with other systems with which the system or system elements must interoperate.

Table 3: *Technical and Technical Management Processes* [14]

shown in Table 4 (from [14]).

The following sections describe these core elements of SoS SE from [9].

Translating SoS Capability Objectives Into High-Level SoS Requirements Over Time

When an SoS is first acknowledged, the SE team is called on to understand and articulate the technical-level expectations for the SoS. SoS objectives are typically couched in terms of needed capabilities, and the systems engineer is responsible for working with the SoS manager and users to translate these into high-level requirements that provide the foundation for the technical planning to evolve the capability over time. To accomplish this, the SoS SE team needs to understand the nature and the dynamics of the SoS to appreciate both the context for SoS expectations and to anticipate areas of the SoS that are most likely to vary in implementation and change over time. The SoS systems engineer has a continuous active role in this ongoing process of translating capability needs into technical requirements and identifying new needs as the situation changes and the SoS evolves.

Understanding the Constituent Systems and Their Relationships Over Time

A key SoS SE activity involves understanding the systems involved in providing the needed SoS capabilities and their relationships and interdependencies as part of the SoS. In an individual system acquisition, the systems engineer is typically able to clearly establish boundaries and interfaces for a new system. In an SoS, systems engineers must gain an understanding of the ensemble of systems that affect the SoS capability and the way they interact and contribute to the capability objectives. Key systems can be outside of the direct control of the SoS management but still have large impacts on the SoS objectives, and it may not be possible to identify all the systems that affect SoS objectives. It is most important to understand the players associated with key systems, their relationships, and their drivers so that options for addressing SoS objectives can be identified and evaluated, and impacts of changes over time can be anticipated and addressed. Understanding the functionality of each system is the basis for understanding (1) how the systems support the SoS objectives, (2) technical details of the systems pertinent to the SoS (e.g., approaches to sharing or exchanging mission information), and (3) the current system development plans, including timing and synchronization considerations. Finally, the SoS systems engineer needs to identify the stakeholders and users of SoS and systems, and understand their organizational context as a foundation for their role in the SoS over time.

Assessing the Extent to Which SoS Performance Meets Capability Objectives Over Time

In an SoS environment, there may be a variety of approaches to addressing objectives. This means that the systems engineer needs to establish metrics and methods for assessing the performance of the SoS capabilities which are independent of alternative implementation approaches. A part of effective mission capability assessment is to identify the most important mission threads and focus the assessment effort on end-to-end performance. Since SoS often comprises fielded suites of systems, feedback on SoS performance may be based on operational experience and issues arising from operational settings. By monitoring performance in the field or in exercise settings, systems engineers can proactively identify and assess areas needing attention, emergent behavior in the

SoS Core Elements	Technical Processes								Technical Management Processes							
	Requirements Development	Logical Analysis	Design Solution	Implementation	Integration	Verification	Validation	Transition	Decision Analysis	Tech Planning	Tech Assess.	Requirements Management	Risk Management	Configuration Management	Data Management	Interface Management
Translating Capability Objectives	X											X	X	X	X	
Understanding Systems and Relationships		X											X	X	X	X
Assessing Performance to Capability Objectives							X		X		X		X		X	
Developing and Evolving an SoS Architecture	X	X	X						X	X		X	X	X	X	X
Monitoring and Assessing Changes									X				X	X	X	X
Addressing Requirements and Solution Options	X		X						X	X		X	X	X	X	X
Orchestrating Upgrades to SoS				X	X	X	X	X	X		X	X	X		X	X

Table 4: *Technical and Technical Management Processes as They Apply to the Core Elements of SoS SE*

SoS, and the impacts of changes in constituent systems on the SoS.

Developing, Evolving, and Maintaining an Architecture for the SoS

Once an SoS systems engineer has clarified the high-level technical objectives of the SoS, identified the systems that are key to SoS objectives, and defined the current performance of the SoS, an architecture overlay for the SoS is developed, beginning with the existing or *de facto* architecture of the SoS. The architecture of an SoS addresses the concept of operations for the SoS and encompasses the functions, relationships, and dependencies of constituent systems, both internal and external. This includes end-to-end functionality and data flow as well as communications. The architecture of the SoS provides the technical framework for assessing changes needed in systems or other options for addressing requirements. In the case of a new system development, the systems engineer can begin with a fresh, unencumbered approach to architecture. However, in an SoS, the systems contributing to the SoS objectives are typically in place when the SoS is established, and the SoS systems engineer needs to consider the current state and plans of the individual systems as important factors in developing an architecture for the SoS. In developing the architecture, the systems engineer identifies options and trades and provides feedback when there are barriers to achieving balance between the SoS and systems.

Monitoring and Assessing Potential Impacts of Changes on SoS Performance

A big part of SoS SE is anticipating change—outside of the SoS span of control—that will impact functionality or performance. This includes internal changes in the constituent systems as well as external demands on the SoS. Because an SoS contains multiple independent systems, the systems engineer must be aware that these systems are evolving independently of the SoS, possibly in ways that could affect the SoS. By understanding the impact of proposed or potential changes, the SoS systems engineer can either intervene to preclude problems or develop strategies to mitigate the impact on the SoS.

Addressing SoS Requirements and Solution Options

An SoS has requirements both at the level of the entity formed by the interoperating constituent systems and at the level of the individual systems themselves. Depending on the circumstances, the SoS systems engineer may have a role at one or both levels. At the SoS level, as with systems, a process is needed to collect, assess, and prioritize user needs, and then evaluate options for addressing these needs. When identifying viable options to address SoS needs, it is key for the systems engineer to understand the individual systems and their technical and organizational context and constraints, and to consider the impact of these options at the systems level. It is the SoS systems engineer's role to work with requirements managers for the individual systems to identify the specific requirements to be addressed by appropriate systems (i.e., to collaboratively derive, decompose, and allocate requirements to systems). This activity is compounded at an SoS level due to the multiple acquisition stakeholders that are engaged in an SoS. The objective is to identify options which balance the needs of the systems and the SoS, since in many cases there may be no clear decision authority across the SoS. Designs for implementing changes to the systems are done by the systems engineers of the systems.

Orchestrating Upgrades to SoS

Once an option for addressing a need has been selected, it is the SoS systems engineer's role to work with the SoS sponsor, the SoS manager, as well as the constituent systems' sponsors, managers, systems engineers, and contractors to fund, plan, contractually enable, facilitate, integrate, and test upgrades to the SoS. The actual changes are made by the consistent systems' owners, but the SoS systems engineer orchestrates the process. The system engineer leads the synchronization, integration, and test across the SoS and provides oversight to ensure that the changes agreed to by the systems are

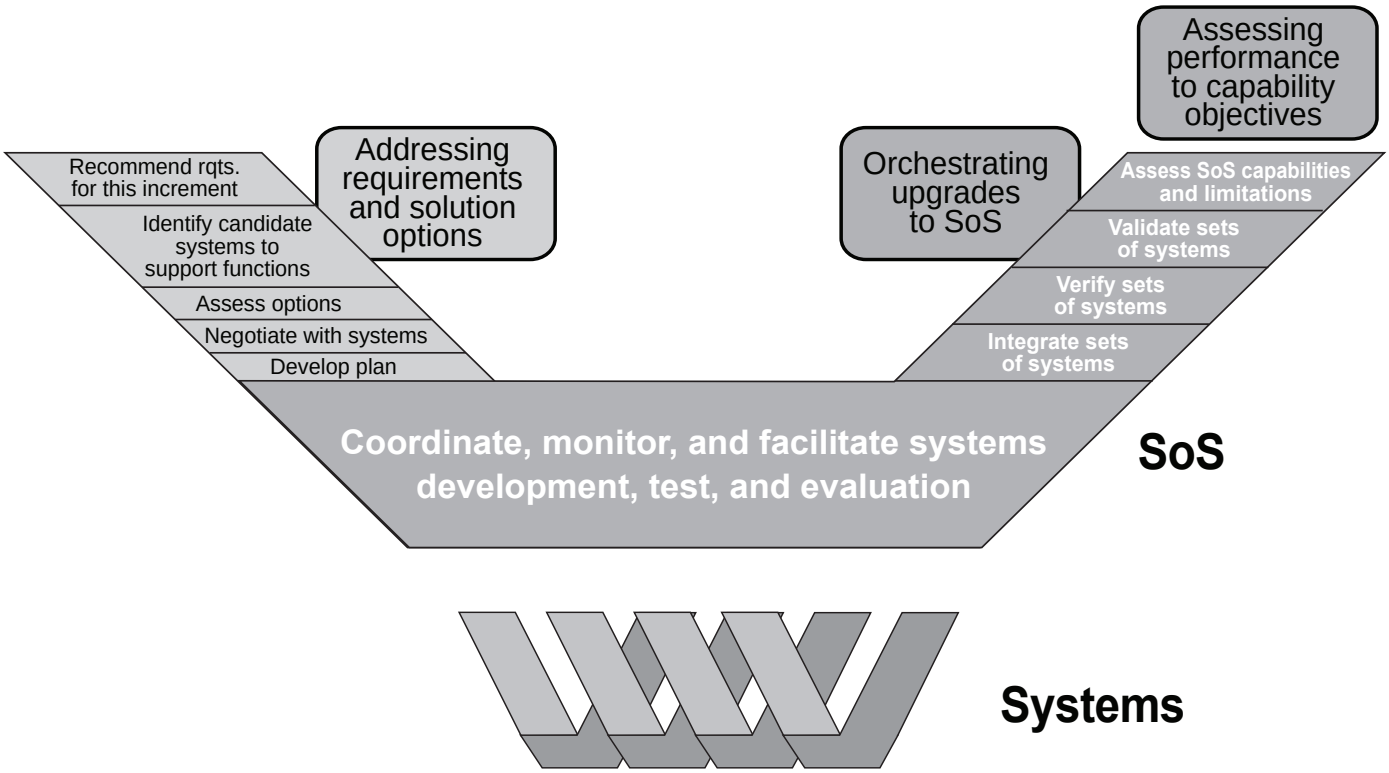


Figure 2: *'Double Vee' Depiction of Upgrades to SoS*

implemented in a way that supports the SoS.

Implementation of *Orchestrating upgrades to SoS*, along with the elements *Addressing requirements and solution options* and *Assessing performance to capability objectives* can be viewed as an extended version of the SE *Double Vee* (see Figure 2); the SoS systems engineer addresses issues across the SoS and the systems engineers of the systems address changes in their systems.

Summary

Systems engineers are increasingly called upon to implement SE for supporting user capabilities in networked environments and are charged with evolving existing and new systems to meet changing user needs. As well, they are challenged to leverage SE processes developed and applied for SE of new systems. In today's SoS environments, individual systems are no longer considered as individual bounded entities, but rather as components in larger and more variable ensembles of interdependent systems, interacting based on end-to-end business processes and networked information exchange to meet user capability needs. Because they are starting with existing systems with independent owners, objectives, and development processes, systems engineers are faced with a new set of conditions for their SE processes. This calls for a new SE framework, reflecting the dynamics and uncertainty of SoS as well as the added complexity of operating in an SoS environment to meet DoD capability needs.◆

References

1. DoD. Quadrennial Defense Review Report. Washington, D.C.: Pentagon, 30 Sept. 2000.
2. Chairman of the Joint Chiefs of Staff (CJCS). CJCS Manual 3170.01C. Operation of the Joint Capabilities Integration and Development System. Washington, D.C.: Pentagon, 1 May 2007.
3. DoD. DODI 5000.2. Operation of the Defense Acquisition System. Washington, D.C.: Pentagon, 12 May 2003.
4. DoD. Quadrennial Defense Review Report. Washington, D.C.: Pentagon, 6 Feb. 2006.
5. Deputy Secretary of Defense. Capability Portfolio Management Test Case Guidance. Washington, D.C.: Pentagon, 14 Sept. 2006.
6. Office of the Under Secretary of Defense for Acquisition, Technology and Logistics (OUSD AT&L) 2004(1). Memorandum on Policy for Systems Engineering in DoD. Washington, D.C.: Pentagon, 20 Feb. 2004.
7. OUSD AT&L 2004(2). Memorandum on Policy Addendum for Systems Engineering. Washington, D.C.: Pentagon, 22 Oct. 2004.
8. OUSD AT&L 2004(3). Implementing System Engineering Plans in DoD Interim Guidance. Washington, D.C.: Pentagon, 30 Mar. 2004.
9. DoD. Defense Acquisition Guidebook. Ch. 4.2.6. "System of Systems Engineering." Washington, D.C.: Pentagon, 14 Oct. 2004.
10. DoD Chief Information Officer (CIO)/Assistant Secretary of Defense for Networks and Information Integration (ASD NII). DoD Net-Centric Data Strategy. Washington, D.C.: Pentagon, 9 May 2003.
11. DoD CIO/ASD NII. Net-Centric Operations and Warfare Reference Model. Vers. 1.1. Washington, D.C.: Pentagon, 17 Nov. 2005.
12. Maier, M. "Architecting Principles for Systems of Systems." Systems Engineering. 1998. Vol. 1, No. 4: 267-284.
13. Dahmann, Judith S., and Kristen Baldwin. Understanding the Current State of U.S. Defense Systems of Systems and the Implications for Systems Engineering. Proc. of Institute of Electrical and Electronics Engineers Systems Conference. Montreal, Canada: 7-10 Apr. 2008.
14. OUSD AT&L. Systems Engineering Guide for Systems of Systems. Washington, D.C.: Pentagon, Aug. 2008 <www.acq.osd.mil/sse/docs/SE-Guide-for-SoS.pdf>.

About the Authors

Judith S. Dahmann, Ph.D., is a senior principal scientist in the MITRE Corporation Center for Acquisition and Systems Analysis, supporting the Director of Systems and Software Engineering U.S. DoD Under Secretary of Defense for AT&L. Prior to this, she was the chief scientist for the Defense Modeling and Simulation Office for the U.S. Director of Defense Research and Engineering (1995-2000), where she led the development of the High-Level Architecture, a general-purpose distributed software architecture for simulations, now an IEEE Standard (1516). Dahmann has a bachelor's degree from Chatham College in Pittsburgh, a master's degree from the University of Chicago, and a doctorate degree from Johns Hopkins University.

The MITRE Corporation
Center for Acquisition and
Systems Analysis
7525 Colshire DR
McLean, VA 22102-7539
E-mail: jdahmann@mitre.org

George Rebovich Jr. is a senior principal engineer at the MITRE Corporation where he has held various systems engineering and management positions. He served in the U.S. Army, including tours of duty in the U.S. and Europe, and as an artillery forward observer with the 101st Airborne Division in Vietnam. Rebovich is a former assistant professor of mathematics at the United States Military Academy (USMA). He holds a bachelor's degree in mathematics from the USMA, a master's degree in mathematics from Rensselaer Polytechnic Institute, a certificate of administration and management from Harvard University, and is a graduate of the U.S. Army Command and General Staff College.

The MITRE Corporation
Advanced Systems Engineering
Center
202 Burlington RD
Bedford, MA 01730-1420
E-mail: grebovic@mitre.org

Jo Ann Lane is currently a principal at the University of Southern California Center for Systems and Software Engineering, conducting research in the area of system of systems engineering. In this capacity, she is currently working on a cost model to estimate the effort associated with system of systems architecture definition and integration. Lane also teaches software engineering courses at San Diego State University. Prior to her current work, she was a key technical member of Science Applications International Corporation's Software and Systems Integration Group, responsible for the development and integration of software-intensive systems and systems of systems.

University of Southern California
Center for Systems and
Software Engineering
941 W. 37th Place
SAL RM 328
Los Angeles, CA 90089-0781
E-mail: jolane@usc.edu

Web Sites

Air Force Operational Test and Evaluation Center (AFOTEC)

www.afotec.af.mil/

Headquartered at Kirtland AFB, New Mexico, AFOTEC is an independent agency responsible for testing new weapons systems for the Air Force, the DoD, and other government agencies. You will find information and news about how AFOTEC supports America's fighting forces with the testing of new weapons systems in realistic battlespace environments, and how they evaluate the capability of these systems to meet warfighter needs.

Systems Engineering Guide for System of Systems (SoS)

www.acq.osd.mil/sse/docs/SE-Guide-for-SoS.pdf

The purpose of this guide, recently released by the DoD, is to address systems engineering (SE) considerations for integrating independently useful systems into a larger system that delivers unique capabilities—an SoS—within the DoD. Drawing from the lessons of current SoS SE practitioners, the guide is intended to provide a resource for systems engineers who are supporting SoS work, particularly as part of an SE team for an SoS.

Quality Is Free

www.philipcrosby.com/25years/

Philip Crosby was a cost-of-quality guru whose many books included the groundbreaking "Quality Is Free," which is still making an impact well after its silver anniversary. At this Web site, learn how Crosby's book has impacted others, learn its main ideas, and get a history lesson on the career and impact of the man *Time* magazine dubbed "the leading evangelist of quality in the U.S."

Web Services (WS) Specifications and Service-Oriented Architecture Interoperability

http://opensource.sys-con.com/node/314083

Just following WS standards and guidelines during the development phase of a project isn't enough to achieve interoperability. This article from *Enterprise Open Source* magazine provides a set of guidelines, insight, and best practices that you can follow to accomplish interoperability when developing WS that make use of specifications across products provided by different vendors, as well as help with specifications that are being developed by different groups.

Interoperability Test and Evaluation: A System of Systems Field Study

Dr. John Colombi, Maj. Brannen C. Cohee, and Maj. Chuck W. Turner
Air Force Institute of Technology

Effective operational test and evaluation (OT&E) is an essential part of successful systems and software engineering. But increased program dependencies, network-centric operations, and growing interoperability requirements have greatly complicated test and evaluation. This article examines the policy, process, and practice of the Air Force (AF) test and evaluation programs, such as Force Development Evaluations (FDEs), particularly during the sustainment of systems. Several observations are made regarding the current process and five areas are emphasized for improvement.

An increasing challenge is facing the OT&E community when operating across multiple weapon systems at various stages of development. After studying the policy, process, and practice associated with an AF Major Command's (MAJCOM's) OT&E program, this article concludes that the AF, while espousing a testing philosophy of *seamless verification*, still needs to transition to a more integrated system of systems (SoS) approach to planning and executing OT&E. Too often, a fielding decision for a single system modification is the goal for smaller test events. This system-centric focus can be misplaced. Indeed, with the dawning of network-centric operations, the SoS imperative is even greater for the successful integration, test, and evaluation of warfighting capabilities. This article highlights several observations and offers some areas for process improvement.

To confirm these challenges, this analysis focused on a geographically dispersed network of ground stations that work with AF and DoD surveillance and reconnaissance (S&R) platforms to provide data, information, and knowledge services for the joint commander and forces in the field. A decade ago, these S&R platforms and their attending ground stations existed in isolation. Since then, however, operational necessities and technological opportunities have birthed a system of increasingly interdependent hardware and software systems, spanning sensors, platforms, data links, communication networks, and software-intensive ground processing resources. The evolution of this SoS has produced remarkable advances in the warfighting capability, but this integration has also created a host of systems engineering (SE) and enterprise management challenges, such as OT&E planning and execution.

The SoS Challenge

The very nature of an SoS makes the enterprise management of traditionally system-centric support processes, such as OT&E, difficult. As Mark Maier points out, even though an SoS operates synergistically, systems in an SoS can operate and are managed independently [1]. This is a premise that is expected to continue into the foreseeable future; therefore, a single organization or program manager will not suddenly take complete managerial control of all systems within the applicable SoS. One reason is that a system may participate in multiple mission threads interacting with a variety of joint organizations and weapon systems. The "Systems Engineering Guide for Systems of Systems" recognizes that an SoS is usually not born of a single development effort but emerges as complex combinations of newly acquired and legacy systems—each with their own management, operations, and support communities—that evolve over time [2]. Annette Krygiel notes that the purpose and capabilities of an SoS change as functions are added, removed, and modified [3]. Compounding the complexity of SoS SE, Pin Chen and Jennie Clothier observe that component systems in an SoS are often systems of systems themselves [4]. All of this complicates the current test and evaluation approaches.

Despite these challenges, operational demands are forcing the AF and DoD to co-evolve historically system-centric processes like OT&E to support the development of SoS and net-centric capabilities [5]. Therefore, to better understand the need for and obstacles to this co-evolution, this study focused on a particular OT&E process and the extent to which it supports an SoS approach. The OT&E process chosen, called an FDE, is managed at the MAJCOM level to make fielding decisions for operational weapon systems as incremental upgrades are made during sustainment. It should be noted that an FDE is one of several types of OT&E called out in AF Instruction (AFI) 99-3, Capability Based Test and Evaluation. Others include Initial Operational Test and Evaluation, Qualification Operational Test and Evaluation, Follow-on Operational Test and Evaluation, Tactics Development and Evaluation, the Weapons System Evaluation Program, Operational Utility Evaluation, Operational Assessments, and Early Operational Assessments.

MAJCOMs conduct FDEs for programs requiring full-rate production or fielding decisions if the AF Operational Test Center chooses not to conduct OT&E. This is typically true for Acquisition Category III programs or maintenance modifications. After a system has been fielded and has entered the sustainment phase of its life cycle, the primary type of test and evaluation used to verify and validate smaller system upgrades is the FDE. As stated in the Air Combat Command instruction, the focus of FDE is a subset of OT&E. FDEs are primarily concerned with sustainment, pre-planned product improvement, as well as tactics, techniques, and procedures development. The objective is to demonstrate the operational effectiveness and suitability of a system as evolutionary upgrades are made to sustain its relevance to the warfighter. In the Air Combat Command (ACC) FDE process, ACC Test Centers (e.g., the AF Warfare Center, the AF Information Operations Center, and the Air National Guard AF Reserve Test Center) are responsible for planning and executing FDEs.

Next, the OT&E process is examined in the context of its governing law and policy. Then, a case study is documented which focuses on a particular test event that involved a networked ground system and one of its airborne partners.

Observations From the Field Study

From Congress, direction for OT&E flows down from four sections of Title 10 of the U.S. Code: Director of OT&E; Survivability Testing and Lethality Testing Required Before Full-Scale Production; OT&E of Defense Acquisition Programs;

and Low-Rate Initial Production of New Systems. The DoD then implements the policies into directives, instructions, and regulations. The AF further clarifies its entire test and evaluation process in the 99-series of departmental instructions, such as AF Policy Directive 99-1, Test and Evaluation Process, and AFI 99-103, Capabilities Based Test and Evaluation [6]. This policy defines the purpose of the AF test and evaluation process and provides a framework for test activities. It also expands on the two major types of tests: Developmental Test and Evaluation and OT&E. The AF philosophy clearly reflects the verification and validation of mission-level capabilities, an emphasis on *seamless verification* across the developmental and operational test activities, the use of an integrated test team (ITT) for test management, and the efficient sharing of test data through a common database. Finally, AF MAJCOMs define operating procedures, as in ACC Instruction (ACCI) 99-101, ACC Test and Evaluation. This instruction further clarifies MAJCOM OT&E procedures. Two examples that stand out in ACCI 99-101 are the Electronic Project Order, for tasking organizations and resources, and the use of a yearly Test Priority List. While extensive, the test policy hierarchy is observed to provide good vision and insightful guidance and establishes a foundation for being able to handle today's SoS test challenges. Several observations on the current process are provided in the following sections.

Seamless Verification Still Has Seams

Through our policy analysis, it was discovered that the AF endorses a capabilities-based concept of seamless verification, especially by mandating the use of ITTs. The ITT consists of a cross-functional group of empowered representatives from multiple disciplines and organizations and is co-chaired by operational testers and the program manager. The problem lies in the meaning of *integrated*. While DoD policy includes both the *horizontal* integration of test and evaluation throughout a system's life cycle and the *vertical* integration of systems under interoperability testing (as shown in Figure 1), the AF sees integrated testing almost exclusively in life-cycle terms [6]. In addition, AF policy also mandates the use of open, shared databases for managing test information. This integrated information management could be how the acquisition and test communities could begin to bridge the vertical seams.

Thus, seamless verification still has seams separating test activities among interdependent weapons systems. AF policy does not mandate organizational structures and management processes that help OT&E organizations conduct their testing activities in an SoS framework. However, the prevalence of systems of systems and the evolution toward net-centricity demands test processes that are both integrated throughout the life cycle of a single weapon system and integrated across entire sets of operational capability.

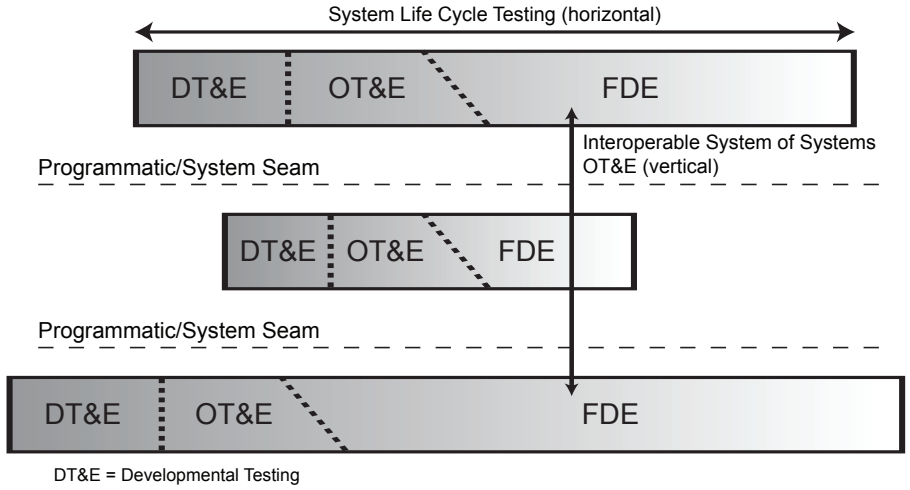


Figure 1: *System Integration During Interoperability Testing*

SoS Approach Not Built In

It was found that the FDE process is flexible in that it can be applied to both system-centric and capabilities-based SoS test events. Even so, it does not include steps to intentionally evaluate SoS capabilities rather than individual systems. Rather, it relies on the insight and foresight of the MAJCOM staff and the test center organizations to properly scope events to approximately demonstrate full warfighting capabilities.

Indeed, the test project manager, generally appointed from one of the MAJCOM's test center organizations, is the central figure in defining the scope of the test. Whether determining the composition of the planning team, developing test objectives, requesting additional support for testing, or developing the test plan, the extent to which FDE demonstrates the operational capability of an SoS depends largely on the vision and initiative of the individual project manager. The MAJCOM's process does not include functions that obligate a project manager to scope a test at the SoS level.

Increasing Load on OT&E

The studied MAJCOM has experienced a dramatic increase in OT&E requirements (from around 200 just five years ago to around 300 today) and the number of short-notice or out-of-cycle requirements (from around 10 percent of the total number of test events five years ago to approximately 40 percent today). The war on terror has certainly contributed to both the number and urgency of OT&E requirements, but one subject matter expert interviewed believes the increased number of acquisition spirals and increments along with the introduction of non-traditional software-intensive weapon systems—such as the networked ground system in our case study—has led to a more expansive, dynamic, and complex environment for MAJCOM-led testing.

Experts and senior leaders have argued that the growing interdependence of systems organized in net-centric architectures will exacerbate the increased load (i.e., the number, complexity, tempo, and expense of test events), calling it exponential growth. With testing resources either remaining static or decreasing over time, this increased load will force the MAJCOM—as well as the broader test community—to develop new methods for testing and evaluating net-centric capabilities.

System-Centric Approach Breaks Down

Our case study starts with an FDE for a modified sensor on board an airborne platform. This sensor modification was needed to support interoperability with a new data link architecture, and it included minor software upgrades to networked ground stations. The MAJCOM assigned the event to its organization responsible for testing modifications to airborne platforms. Understanding the organic relationship between the platform and the ground system, test planners knew they needed to demonstrate the end-to-end interoperability of four interdependent

systems: the sensor, airborne platform, data link, and ground station. Yet, the test was not planned as a demonstration of the operational effectiveness and suitability of an SoS capability, but as a validation of the single-sensor system with the support of these other contributing systems. While this may seem like a subtle distinction, it led to a variety of coordination and communication breakdowns throughout test planning. In retrospect, the FDE and subsequent fielding decision should have been for the combined SoS, made up of the sensor, airborne platform, data link, and ground stations, which would have necessarily involved a broader cross-section of stakeholders from the genesis of the test event. When SoS thinking is not *baked into* the overall test and evaluation process, even highly interdependent systems will have difficulty coordinating test events that are effective and relevant for the constituent systems and the SoS as a whole.

Recommendations for Net-Centric OT&E

Though important efforts have been made at all levels to promote SoS-level testing, the default focus of testing is still on individual systems as opposed to whole capabilities. As net-centric operations mature, this approach will have to change. Indeed, as our field study indicates, the DoD is already feeling the pressure that was predicted nine years ago:

> Testing systems will become far more complex since the focus will not be on the performance of individual systems, but on the performance of federations of systems. [7]

At the National Defense Industrial Association Test and Evaluation Summit in 2004, DoD officials elaborated on the net-centric challenges to traditional, system-centric testing [8]:

- The shifting focus from platforms to capabilities and SoS solutions.
- The increasing complexity of systems combined with increasing interdependencies among systems of systems.
- The increasing operational demand for broader and deeper integration among disparate systems.
- The exponential growth in functional and physical interfaces introduced by the proliferation of network participants (both newly developed and legacy).
- The increased requirements for test and evaluation initiated by the evolutionary acquisition philosophy of build-a-little, test-a-little.

As the heralds of net-centricity emphasize, the DoD's transition from Industrial Age (platform-centric) to Information Age (net-centric) operations must include a co-evolution of supporting processes. Testing is one of those supporting processes that must co-evolve with technology. The following recommendations are believed to best improve MAJCOM-level OT&E, but should be extensible to the AF and DoD OT&E. These can be considered attributes of a future process; while not meant to be exhaustive, they provide a starting point for continuing research on how to evolve today's process to complement a more interoperable net-centric environment.

Scope OT&E Events at the SoS Level

This article advocates an SoS (instead of a system-centric) approach to OT&E. However, the question arises of how to appropriately scope the boundaries of SoS test events. This is where the DoD AF and AF Enterprise Architecture (EA) could offer practical help. As the use of EAs continues maturing, they will provide effective models for assessing how weapon systems can and should interoperate in order to provide warfighting capabilities to the joint commander. These models will help planners define the boundaries of an SoS, and they will document what the MITRE Corporation's Prem Jain calls a *mission thread*:

> A precise, objective, description of an important task … a time-ordered operational event diagram that captures discrete, definable, interactions among human operators and/or technological components. [9]

Jain argues that these mission threads will support modeling and simulation (M&S) activities for net-centric test and evaluation. In the same way the AF uses high-fidelity simulators to slash the costs associated with training its pilots, the test community could use mission thread-based M&S to validate SoS and net-centric capabilities at a fraction of the cost, time, and operational impact incurred by live, end-to-end tests.

Validate SoS Interoperability

By advocating SoS testing, an endless web of end-to-end interoperability tests is not envisioned with every other possible weapon system and every configuration. Rather, changes to individual weapon systems should be evaluated according to net-readiness criteria to validate their interoperability with the rest of the SoS or net-centric enterprise. This does not mean just checking to see if a modified weapon system is IP-enabled. Net-readiness is a comprehensive concept that implies interoperability at many layers of the communications hierarchy and beyond: physical, logical, syntactic, and semantic [2]. For nearly 30 years, both government and industry have actively explored research on interoperability measurement with the goal of creating a straightforward way of measuring, reporting, and then improving the interoperability of complex networks of people, equipment, processes, and organizations. Researchers have used more than 30 definitions of interoperability and have documented more than 60 distinct types of interoperability, numerous interoperability attributes, and 14 foundational interoperability measurement models and methodologies [10].

For SoS testing, a set of net-readiness objectives (see Table 1) based on the DoD's Net-Centric Data Strategy [11] is proposed. Incorporating these objectives into SoS events would ensure that modified systems continue to conform—at their interface with the network—to the convergence protocols specified in the net-centric architecture. Instead of evaluating all end-to-end relationships to validate interoperability within the SoS, a test event could confirm the integrity of the SoS simply by demonstrating the modified system's adherence to the network's convergence protocols. This technique would greatly reduce the test load on OT&E organizations while simultaneously allowing them to conduct evaluations focused on the operational effectiveness of the net-centric SoS, as opposed to the individual systems within that SoS.

Prioritize FDEs According to Operational Risk

Although a simulation and net-readiness demonstration at the network interface will help mitigate the test load associated with SoS and net-centric operations, decision makers will still expect a certain level of live, end-to-end testing in realistic scenarios to validate higher-risk capabilities. There will always be too much to test, forcing the operational and test communities to develop a reasonable means of prioritizing test events. Complicating this issue is a fundamental property of net-centric operations: New transactions, interdependencies, missions, and capabilities as additional (even unanticipated) sensor, shooter, and command and control nodes join the network. The very nature

of net-centric operations implies that the DoD will never completely anticipate all of the relevant operational nodes and precisely how those nodes will interoperate to accomplish a mission.

Thus, the crucial criterion for prioritizing and scoping SoS OT&E events is operational risk. For low-risk development or sustainment efforts, operational decision makers may need to be satisfied with developmental test results validated in an M&S-based operational test. For medium-risk projects, testers may use a synthetic test strategy that employs a small number of distributed operational events in an M&S framework. The test and evaluation community may need to reserve traditional end-to-end events for the highest risk efforts. The key will be for operational decision makers to set an appropriate risk threshold for each development or sustainment program and seek the best value test option in terms of time, money, and testing/operational resources to achieve that threshold.

Focus on Interfaces

Without trivializing the complexities of an SoS, perhaps more emphasis may have to be placed on the definition, development, and test and evaluation of interfaces. Interfaces and interface management have always been an important aspect of SE. Maier and Rechtin state this design heuristic: "The greatest leverage in system architecting is at the interfaces … the greatest dangers are also at the interfaces" [12]. According to the "Defense Acquisition Guidebook," this heuristic can be an interface that is:

> The [logical], functional, and physical characteristics required to exist at a common boundary or connection between persons, between systems, or between persons and systems. [13]

During an interface control document (ICD) review of one space program, we examined the impact of interface design and development. Over a three-year period following contract award, 596 program engineering items were examined. These items included: requirements changes, specification updates and clarifications, ICD changes, SE management documents, baseline schedule changes, test plans, and proposed risk-mitigation actions; ICD-related actions comprised 190 (or one-third) of the total number of actions. A second aspect of interface management was in contract costs. From a set of 77 contractual modifications after critical design review, 43 (nearly 56 percent) were in some way related to interfaces either through studies, ICD updates, or implementations of necessary requirement changes. More interestingly, ICD-related issues resulted in nearly $31.5 million (or 44 percent) of the cost impact to this program. Although this is just one example, it is an indication of interface management challenges during development and clearly represent an area that will require similar effort during OT&E.

Unfortunately, interface management alone may be insufficient to understand the complexity within an SoS. For network-centric information systems, one must examine the internal properties and behaviors of the logically connected systems and their users who find, fuse, modify, and ultimately use shared data.

Objective	Description
Visible	Users and applications can discover the existence of data assets through catalogs, registries, and other search services. All data assets (intelligence, non-intelligence, raw, and processed) are advertised or *made visible* by providing metadata, which describes the asset.
Accessible	Users and applications post data to a *shared space*. Posting data implies that: (1) descriptive information about the asset (metadata) has been provided to a catalog that is visible to the enterprise and (2) the data is stored such that users and applications in the enterprise can access it. Data assets are made available to any user or application except when limited by policy, regulation, or security.
Understandable	Users and applications can comprehend the data, both structurally and semantically, and readily determine how the data may be used for their specific needs.
Trusted	Users and applications can determine and assess the authority of the source because the pedigree, security level, and access control level of each data asset is known and available.
Agile	Many-to-many exchanges of data occur between systems, through interfaces that are sometimes predefined or sometimes unanticipated. Metadata is available to allow mediation or translation of data between interfaces, as needed.
Responsive	Perspectives of users, whether data consumers or data producers, are incorporated into data approaches via continual feedback to ensure satisfaction.

Table 1: *A Template for Net-Readiness Testing Objectives*

Employ Integration Environments

The heavy use of M&S and synthetic testing to supplement traditional test and evaluation presupposes the use of integration environments to build SoS and net-centric operational capabilities. Annette Krygiel calls an integration environment:

> … a concept, not an organization. It is the environment of people, processes, and infrastructure used by a team consisting of acquisition and operational personnel to manage the integration before the product is deployed for an operation or an experiment and to sustain it afterward. [3]

Thus, an integration environment is a concept that transcends not just OT&E but is an essential SE infrastructure for the early and continuous integration of testing efforts in SoS development and sustainment. Thus, an integration environment is critical in squeezing the most overall value from OT&E and in reducing the amount of effort required late in the development cycle [14].

Clearly, integrated databases open to all test and evaluation stakeholders are just the tip of the iceberg in terms of the tools needed to achieve seamless verification. Employing integration environments would allow test teams to achieve synergy throughout the life cycle of component systems and across the networked SoS. It would allow early, comprehensive, and ubiquitous test and evaluation throughout the development of SoS capabilities, reducing the test load on test center organizations and giving them the freedom to focus on adding value where it counts for MAJCOM decision makers: in reducing the operational risk of fielding and employing warfighting capabilities.

Summary

This field study of the policy, process, and practice of FDE concludes that the testing community must shift from system-centric testing to a more SoS approach. The

DoD's ongoing transformation to net-centric operations makes the co-evolution of SoS test and evaluation an even greater imperative. While the five areas for improvement were derived from MAJCOM FDE practice, they reflect similar concepts for more formal OT&E. Improving the operational realism of network-centric environments, ensuring timely performance of operational information, and facilitating adequate test and evaluation resources continue to be priorities of the OT&E director [15]. Likewise, AF OT&E continues to be challenged with SoS test planning and execution. The software engineering community must continue to design and test capabilities with an SoS focus and develop advanced modeling and simulation capabilities to enable affordable SoS testing in a net-centric environment. Likewise, the test community, while emphasizing seamless verification, needs to make continued progress in capabilities-based interoperability testing across information-intensive SoS.◆

References

1. Maier, Mark W. "Architecting Principles for Systems of Systems." Systems Engineering. 1.4 (1998): 267-284.
2. Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics. Systems Engineering Guide for System of Systems. Vers. 1.0. Aug. 2008 <www.acq.osd.mil/sse/docs/SE-Guide-for-SoS.pdf>.
3. Krygiel, Annette. Behind the Wizard's Curtain. Washington, D.C.: DoD Command and Control Research Program. 1999 <www.dodccrp.org/files/Krygiel_Wizards.pdf>.
4. Chen, Pin, and Jennie Clothier. "Advancing Systems Engineering for Systems-of-Systems Challenges." Systems Engineering. 6.3 (2003): 170-183.
5. Alberts, David S. Information Age Transformation. Washington, D.C.: DoD Command and Control Research Program. 2002 <www.dodccrp.org/files/Alberts_IAT.pdf>.
6. USAF. "Capabilities Based Test and Evaluation." Air Force Instruction 99-103. Washington, D.C.: USAF, 6 Aug. 2004.
7. Alberts, David S., et. al. Network Centric Warfare. Washington, D.C.: DoD Command and Control Research Program. Aug. 1999 <www.dodccrp.org/files/Alberts_NCW.pdf>.
8. Lamartin, Glenn F. The Role of T&E in the Systems Engineering Process. Proc. of the National Defense Industrial Association T&E Summit. Washington, D.C.: 17 Aug. 2004 <http://proceedings.ndia.org/487F/lamartin.pdf>.
9. Jain, Prem. "Mission-Model Driven Process: Test and Evaluate Net Centric Capabilities." The MITRE Corporation. Technical Paper. 2007.
10. Ford, Thomas C., et al. Survey on Interoperability Measurement. Proc. of the 12th Annual International Command and Control Research and Technology Symposium. 2007 <www.dodccrp.org/events/12th_ICCRTS/CD/html/papers/096.pdf>.
11. Stenbit, John F. DoD Net-Centric Data Strategy. 9 May 2003 <www.defenselink.mil/cio-nii/docs/Net-Centric-Data-Strategy-2003-05-092.pdf>.
12. Maier, Mark W., and Eberhardt Rechtin. The Art of Systems Architecting. 2nd ed. New York: CRC Press LLC, 2002.
13. DoD. Defense Acquisition Guidebook. 12 Dec. 2004 <https://akss.dau.mil/dag>.
14. Brown, C. David. Transformation of Army Test and Evaluation. National Defense Industrial Association T&E Summit, Washington, D.C. 17 Aug. 2004 <http://proceedings.ndia.org/487f/brown.pdf>.
15. McQueary, Charles. Proc. of National Defense Industrial Association SE Conference. 23 Oct. 2007 <www.dtic.mil/ndia/2007systems/SEGS/McQueary. pdf>.

About the Authors

John Colombi, Ph.D., is an assistant professor of SE at the AF Institute of Technology (AFIT). He teaches graduate courses and leads sponsored research in support of the SE program. Retiring after 21 years in the AF, Colombi led Command, Control, Communications, Computer (C4), Intelligence, and Reconnaissance systems integration activities including SE for the Airborne Warning and Control System at Hanscom AFB.

Dept. of Systems and Engineering Management
AFIT/ENV
Wright-Patterson AFB, OH 45433
Phone: (937) 255-3355 ext. 3347
Fax: (937) 255-4981
E-mail: john.colombi@afit.edu

Maj. Brannen C. Cohee is director of operations at the 315th Training Squadron at Goodfellow AFB, Texas. He supervises the training of AF-enlisted and officer intelligence specialists. He is currently deployed as the intelligence planner with the Air Component Coordination Element in Kabul, Afghanistan. Cohee received his commission from the USAF Academy and a master's degree in public policy from Harvard University before entering intelligence officer training.

315th Training Squadron
154 Canberra ST
Goodfellow AFB, TX 76908-4002
Phone: (325) 654-5649
DSN 477-5649
E-mail: brannen.cohee@goodfellow.af.mil

Maj. Charles "Chuck" W. Turner is chief of the Current Operations Section for the U.S. Central Command's C4 (J6) Directorate. He recently graduated from AFIT, completing a course of study in C4 and Information Systems with a special emphasis in cyber defense. As a communications computer officer, Turner has served in a variety of operational, support, and staff positions involving communications systems.

HQ CENTCOM/CCJ6-CO
7115 S Boundary BLVD
MacDill AFB, FL 33621-5101
Phone: (813) 827-6458
Fax: (813) 827-2211
E-mail: turnerc@centcom.mil

Key Transformational Techniques to Achieve Enterprise-Scale Interoperability

Shamlan Siddiqi
Keane, Inc.

This article examines key modernization and transformation strategies for interoperability, including enterprise use of open source, service-oriented architecture (SOA), and agile techniques in software development. The article concludes with a real-world case study on legacy modernization and interoperability for a major government agency through use of these tools and techniques.

Modern concepts such as SOA and principles around Web 2.0 and Web 3.0 rely on interoperability as their foundation. This is different from the traditional view of integration where one or more adapters are required to glue things together. The intelligent use of standard interfaces between multiple components, leveraging open source technologies, sophisticated SOA design principles, and a flexible software development methodology can be a much more cost-effective, efficient, and scalable option for the enterprise interoperability of systems.

While the adoption of open source technologies for interoperability has been growing at a fast pace over the last few years, I find that there is still apprehension in the *enterprise use* of open source as the dominant platform in the interoperability of mission-critical systems. By *enterprise use*, I mean using a complete stack of non-proprietary open source tools for everything including a back-end database at the data layer, core applications and modules at the application layer, a messaging engine/service bus or standard interface strategy at the integration layer, and creative user interface design techniques leveraging advances in the Web 2.0 space. That being said, open source software by its very nature requires that its software code be open, extensible, and openly available. This paradigm enables programmers globally to share thoughts and ideas, constantly increasing the power and quality of development tools and large applications. Additionally, I believe that the rapid adoption of SOA design principles is a significant boost to the *modernization* of interoperable open source tools. Of course, this rapid evolution of sorts creates problems.

Issues With Interoperability and Security in the Open Source Arena

With new tools and technologies being released often, agencies face challenges in determining whether a particular tool will work with an existing legacy component, whether the operating system is compatible, and if the database will work. There are also the myriad complexities that come with defining the proper interface to create reusability. Some major interoperability issues include centralized identity management, data integration including real-time data synchronization, batch transfer, and portability. This is critical as organizations want their solutions to work across different platforms, such as the various Linux options as well as Windows.

Other issues include the perception of security (or lack thereof) with open source tools as well as the trustworthiness of the data. It is easy for developers to be able to add malicious code or add functions (etc.) to source code they have downloaded. Both of these challenges can be mitigated by the appropriate selection of open source projects. It is advisable to select projects that are more established, come from a reliable source, and have built-in security mechanisms rather than ones that may not have huge support or documentation. Typically, source code from reliable projects does not usually have too many security issues because they are controlled by an open source committee or group. Many development projects use open source tools and technologies and provide ongoing maintenance and support on those projects as well. In fact, based on cases I have encountered, support and maintenance issues are resolved more rapidly with open source because a developer can go into the code and make the fix. The alternative—waiting for a product vendor to come in, assess the problem, and make adjustments—usually takes a significant amount of valuable development time and money.

> *"This paradigm enables programmers globally to share thoughts and ideas, constantly increasing the power and quality of development tools and large applications."*

Interoperability Using an SOA Design and Enabled by Web Services (WS) and the WS* Protocols

Using the SOA design paradigm, interoperability is typically accomplished by developing WS using industry standard programming languages such as eXtensible Markup Language (XML), Web Services Description Language (WSDL), and others. These basic WS standards have evolved over the last few years and the Web Services Interoperability Organization (WS-I)—an open industry organization chartered to establish best practices for WS interoperability—has released Basic Profile (BP), containing implementation guidelines for basic WS standards. Many open source tools are starting to adopt these standards using guidance from BP as part of their ongoing product road map. I think this is a critical success criterion in the further evolution and adoption of open source tools for enterprise integration and other enterprise-level functions. However, just following WS-I standards and guidelines is not sufficient enough to achieve interoperability.

Best Practices in Implementing WS-Enabled SOA Interoperability

It is important to note that the WS*-based development can be complex. Based on experience (and as typically in many things), an incremental strategy is the most effective one. Start implementing using only the basic WS specification as opposed to multiple WS* protocols, which can add more complexity to development. This will create problems when trou-

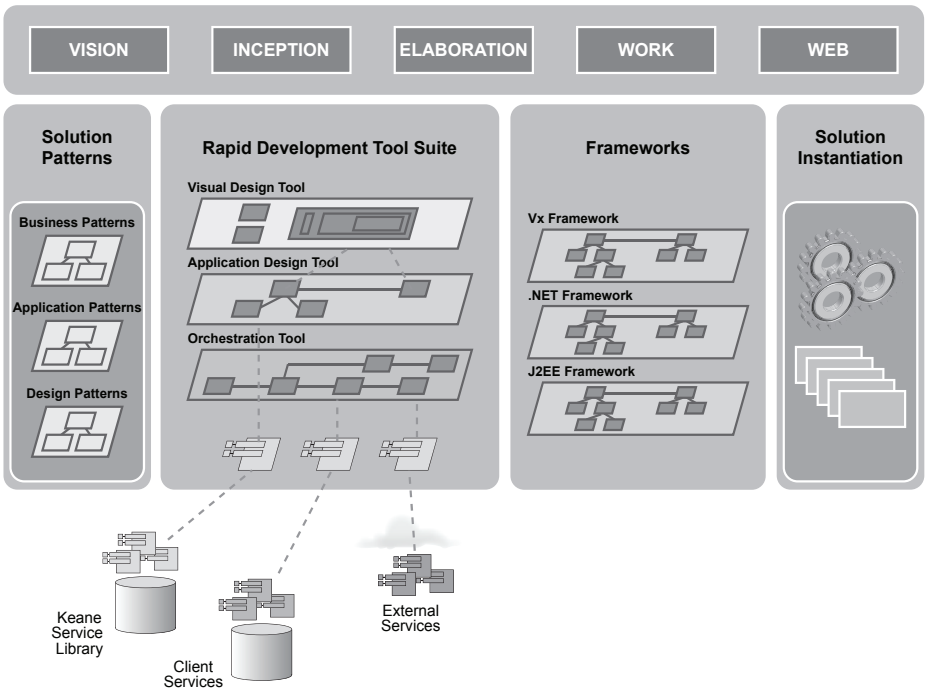


Figure 1: *Agile-RUP Methodology*

bleshooting. Many government organizations are facing major development hurdles because of large-scale implementations of WS without using a chunking strategy. These days, vendors provide BP-compliant products and have tested those products comprehensively for interoperability. It is important to follow these guidelines and use these tools. The open source community (as previously mentioned) is adopting these specifications for interoperability as well. Additionally, it is always important to seek advice from multiple forums and blogs, as there are people who may have encountered similar issues.

The BP consists of guidelines or best practices recommending how the mentioned specifications (XML, WSDL, etc.) should be used together to develop interoperable WS. The guidelines cover messaging, description, discovery, and security. Some key Web service specifications covered in the initial release (BP 1.0) include common standards such as XML 1.0 and WSDL 1.1[1].

When discussing the importance of interoperability and standards, I like using the example of Boeing. When taking apart old 747s, Boeing reuses a lot of the core parts (recovering more than $6.8 million) due to the use of common standards in the design of the original Boeing 747. However, specific standards must be agreed upon before this reuse can take place, and this is not applicable to other jets. Similarly, for two applications to communicate, they must also agree on the specific data standards to be encoded, such as using XML; this, unfortunately, does not always happen due to conflicting specifications adopted by different vendors and agencies. A case in point is the proposed reliable-network protocol that identifies, manages, and tracks the reliable delivery of messages between source and destination WS. According to [1], there are currently two competing specifications: WS-Reliable Messaging (using a specification supported by IBM, Microsoft, BEA, and TIBCO) versus WS-Reliability (promoted by Oracle, Sun, Fujitsu, Hitachi, and Sonic Software, among others). The open source world is starting to adopt these standards feverishly in the hopes of expanding their imminent presence in the larger software market.

Case Study

The power of these open source tools and the low cost of development are precisely why Keane, a billion-dollar global business and IT consulting firm, went with a predominantly open source model for the design and development of the USAF Logistics Systems.

Through both technology and business insight, the project has been successful in its use of cost-effective interoperable open source tools to help the USAF achieve its mission and goals. The system handles 5,000 users and around 3,000 simultaneous users during peak time. The project is saving around 2 million lines of code by using a composite commercial off-the-shelf approach and utilizing an intelligent mix of open source tools. This cost-effective combination provides the USAF with the interoperability and visibility of assets across both retail and wholesale systems. It also integrates the information in near real-time into an intuitive, Web-based interface using Google Web Tools to enhance the Web 2.0 capability. The project also used an SOA-based approach to integration, leveraging the latest in open source libraries and adopting standards such as XML and the Java platform; these rapidly delivered interoperability with numerous systems and exposed legacy systems without modification.

Some core and notable open source technologies used included the Spring Framework, an application and integration framework for the Java platform; iBatis for persistence; Apache Axis, an XML-based Web service framework application development done in Java using Tomcat and extensive use of open source libraries; and Java Enterprise Edition/Java 2 Enterprise Edition components. Using these technologies to develop an innovative transformation and business rules engine is an intelligent *enterprise use* of open source technologies to achieve interoperability.

Enterprise Software Development Methodology – the Blended Agile-Rational Unified Process

For the USAF Logistics Systems, a blended approach to enterprise application development combining agile and VIEWW[2] methodologies was used to quickly deliver modernized solutions meeting 100 percent of client expectations. This collaborative development approach allowed the team to more efficiently review and test, thereby helping to speed development efforts. The Agile-Rational Unified Process (RUP) methodology is beneficial because it works with poorly understood architectures, produces a highly reliable system, produces a system with large growth capability, manages risks, can be constrained to a predefined schedule, provides management with progress visibility, and allows for in-process corrections. Figure 1 depicts the Agile-RUP methodology in some detail.

Through utilizing open source tools and innovative forward-thinking solutions, government best practices can be used as examples to follow in the commercial arena. Served industries—such as defense and aerospace, pharmaceutical, manufacturing, and financial—could benefit greatly from efficient and cost-effective enterprise-level supply chain systems, collaborative decision support engines, and intel-

ligent rules agents that change as the global economy changes.

Conclusion

I believe that the adoption of interoperable open source tools using SOA design principles will drive the future development of innovative forward-thinking solutions. Government best practices will be used as examples to follow in the commercial arena. Agencies supporting defense and intelligence—as well as the broader federal health care and financial arenas—will benefit greatly from efficient and cost-effective interoperable systems at the enterprise-level.◆

Notes

1. Learn more about BP 1.0 at WS-I: <www.ws-i.org/>.
2. VIEWW stands for the different phases of the development life cycle: Vision, Inception, Elaboration, Work, and Web. It is Keane's equivalent to the RUP.

References

1. Narang, Sanjay. "Web Service Specifications and SOA Interoperability." Enterprise Open Source Magazine. 11 Feb. 2007 <http://opensource.sys-con.com/node/314083>.

Additional Reading

1. Smith, Roger. "Using Open Source Tools to Build Interoperable Web Services." 8 Sept. 2004 <www.linux.com/feature/38651>.
2. Krafzig, Dirk, Karl Banke, and Dirk Slama. Enterprise SOA – SOA Best Practices. Indianapolis, IN: Prentice Hall PTR, 2005.
3. "Interoperability Still Stumbling Block for OpenSource in 2008." eWeek.com. 3 Jan. 2008 <www.eweek.com/c/a/Linux-and-Open-Source/Interoperability-Still-Stumbling-Block-for-Open-Source-in-2008/>.

About the Author

Shamlan Siddiqi serves as director of enterprise integration at Keane, Inc. He has more than 10 years of consulting experience working on and managing all phases of both the pre-sales and systems life cycle. At Keane, he leads the delivery of business and technical solutions to clients at all levels of the enterprise. Siddiqi has successfully led complex SOA and enterprise architecture projects in global health, the government/public sector, and for Keane's global financial services.

Keane, Inc.
1410 Spring Hill RD
STE 500
McLean, VA 22102

Modeling and Analysis of Interoperability Risk in Systems of Systems Environments

William B. Anderson and Philip Boxer
Software Engineering Institute

This article describes the use of a set of modeling and analysis techniques in an interoperability risk probe that found gaps in the ability of a North Atlantic Treaty Organization (NATO) modernization program to react to changing demands. The modeling and analysis techniques were used to create models of the people, processes, and technologies of the program and to represent the way demands were placed on this complex socio-technical system. Analysis of the models revealed interoperability risks that were manifested in the linkages between operational requirements of functional capabilities and the way in which those capabilities were being maintained. The risks identified in this probe were typed as mission, composition, and performance risks. The structural models produced by the techniques bring a welcome engineering rigor to the process of examining interoperability.

A National Defense Industrial Association report suggests that modeling can aid the DoD throughout the system of systems (SoS) development life cycle [1]. Model-based dynamic system analysis provides insight into otherwise unobservable dimensions that can help characterize an SoS. One of those dimensions is interoperability risk, which is manifested in the linkages between operational requirements of functional capabilities and the way in which those capabilities are maintained.

This article describes a model-based interoperability risk probe[1] of a NATO modernization program. Using a rapid assessment engagement format, the Software Engineering Institute (SEI) modeled the NATO program as a system of social and technical systems. The probe involved workshops and interviews conducted over a two-week period, followed by analysis of the data gathered. In the probe, we interpreted SoS and interoperability in a broad sense. We examined the hardware and software in the context of its operational and sustainment environments. Therefore, the SoS examination included the many ground and airborne systems and the diverse organizations (social systems) required to operate and sustain the NATO program. In this article, we are emphasizing the modeling and analysis techniques employed over the specific details of the case, because those details are confidential to NATO[2].

The risk probe emphasized the importance of demand on a systems of systems environment. If demand is stable and pre-identified—large nation-state military threat scenarios, a huge and stable demand for sport utility vehicles, or the best health care that money can buy—traditional hierarchical structures and monolithic systems work well. However, demand conditions can change—terrorism has redefined military threats, ever-increasing gasoline prices have affected consumer choice in automotive vehicles, and health care costs have skyrocketed—forcing market-driven demand responses from systems of systems.

"If demand is stable and pre-identified ... traditional hierarchical structures and monolithic systems work well. However, conditions changed ... forcing market-driven demand responses from systems of systems."

An assumption underlying the techniques is that interoperability issues are—and should be—strongly influenced by the need/desire to be reactive to changing demands. As a result, these modeling and analysis techniques find gaps in an organization's ability to react to changing demands. The goal is to model complex emergent patterns of behavior (and gaps therein) that are not directly intended by any single governance entity within a complex SoS. The techniques model physical and social aspects of enterprises associated with the conception, construction, fielding, operations, and evolution of complex systems and systems of systems. An enterprise can include multiple organizational entities, often under different management and ownership structures. The techniques model the roles and interrelationships of the enterprise's physical and social elements across organizational entities and their ability to form, use, and evolve automated systems within the systems' *operational context of use*—both today and for the future.

The techniques probe three general categories of risk:

1. **Performance.** The risk that subsystems within system elements will not interoperate in the ways needed to respond to demand.
2. **Composition.** The risk that a set of systems that need to interoperate within a given SoS cannot be made to interoperate in the ways being demanded of them.
3. **Mission.** The risk that an SoS will not function within its operational context of use in the ways demanded of it.

Table 1: *Perspectives Represented in the Workshops*

Client Perspective	Description
Physical	The physical realization of a complex system or SoS within its operational context-of-use.
Cognitive	The knowledge associated with the acquiring, building, or evolving of a complex system or SoS.
Effects-based	Mission or business effects, current and future, that the capabilities provided should support.

NATO Interoperability Probe Approach

In the NATO probe, we used workshops and small group interviews to examine

interoperability at six successively broader levels that form a stratification:

1. Services, systems, and know-how.
2. Activity chains involved in integrating components.
3. Activities supporting the operational capability.
4. Orchestration of capabilities by a crew and operators.
5. Operational performance of the capability.
6. Mission environments.

At the broadest level—mission environments—we sought interoperability risks in the way different command authorities were able to collaborate. At narrower levels, we looked at the way different Command and Control Information Systems assets and capabilities produced combined effects. At the narrowest levels, we examined the ability of hardware, software, and firmware to work together as effective subsystems within larger systems.

We built models of the way these levels interoperated in terms of the relationships between people, processes, technologies, and operational demands associated with all aspects of the formation, use, and evolution of the NATO SoS. The interoperability models were produced in stages as described in the next three sections:

1. **Visual Model Representations.** Layered, graphical depictions that conformed to a specific syntax of symbols and interconnection rules. This stage was interactive; subject matter experts worked with the modelers in a workshop setting.
2. **Model Matrices.** A set of stratified spreadsheets that juxtaposed activities and events with mission environments. This stage was generated offline by the study team from the visual model representations (Figure 1).
3. **Interoperability Landscapes.** The interrelationships specified in the matrices. This stage was generated offline and became the primary reasoning representation back to the stakeholder community (Figures 2-5).

Visual Model Representations

Through interviews and three workshops (each workshop focused on one of the three client perspectives listed in Table 1), the SEI team and the client representatives created models to ensure that the perspectives of all relevant stakeholders were represented.

To initiate the modeling, we used a brainstorming aide, referred to as *Stakeholder Collaboration Analysis* or the *4-Colors* for short, which has origins in war gaming. It facilitated discussion and initial expression of the dynamic characteristics of the social and technical systems being examined. In war games, blue represents friendly forces; red, the enemy forces; white, the referees; and black, intelligence. We modified this rubric to fit the NATO context.

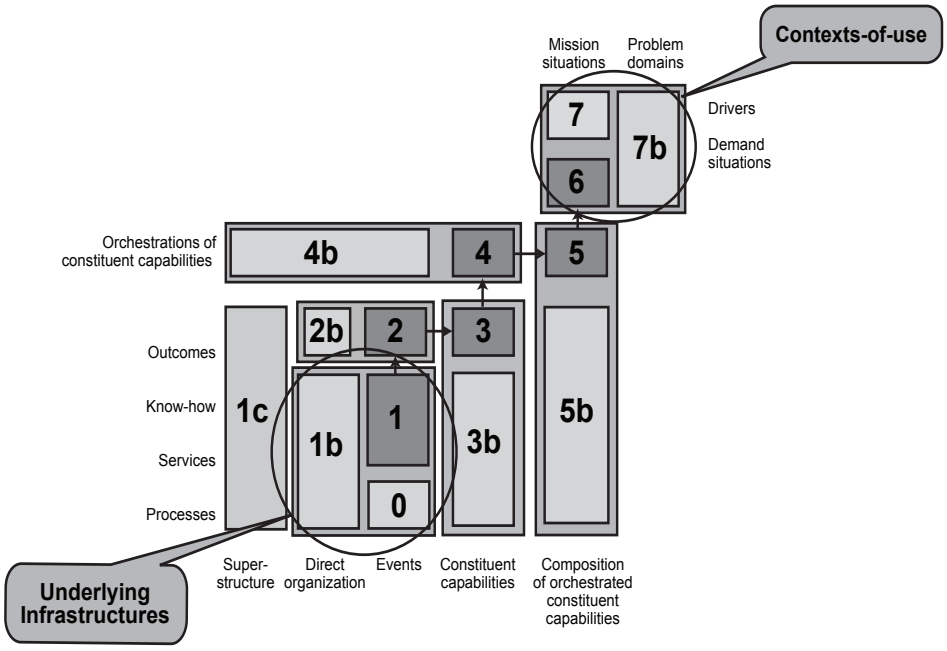


Figure 1: *Matrix Stratification With Exemplar Entities*

In NATO's case, we applied the colors to describe the program's capabilities (blue) in relation to the particular demands being placed upon them (red), within the context of what is driving the mission environment (black). White was used to represent the management of the interoperability among all of these constituents. A significant study team hypoth-

Figure 2: *Interoperability Landscape*

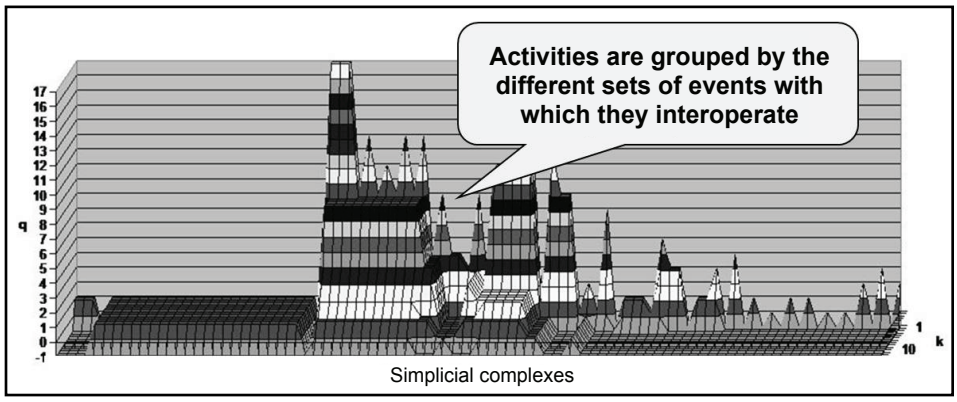
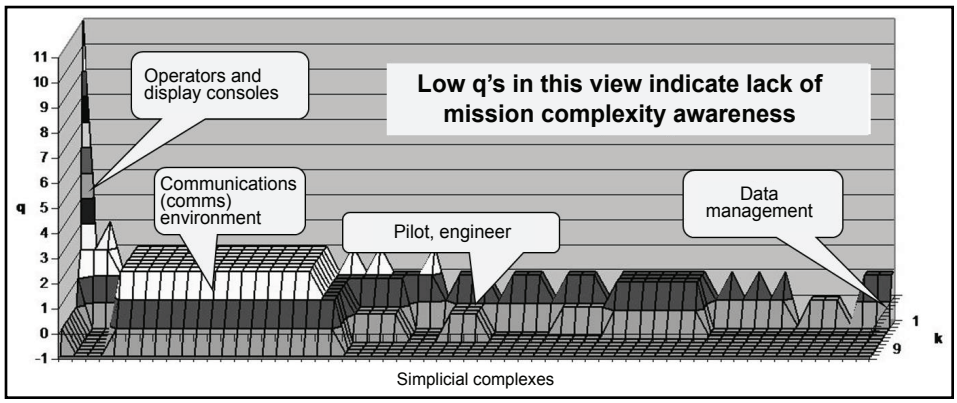


Figure 3: *Mission Awareness Landscape*



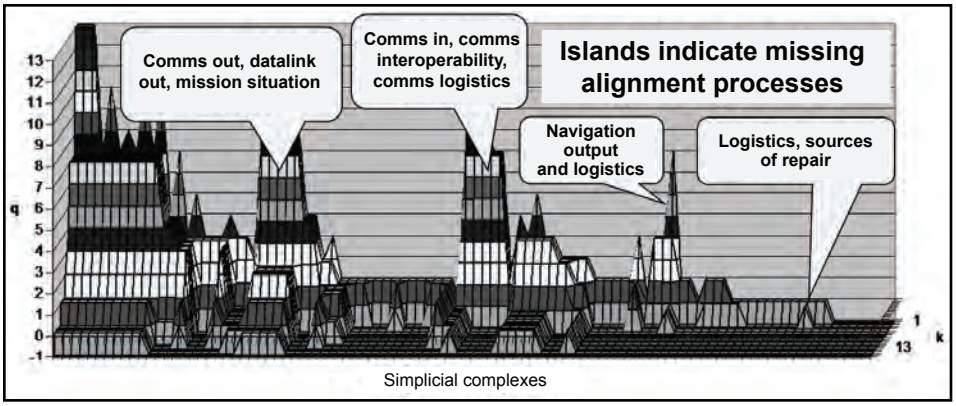


Figure 4: *Orchestration Landscape*

esis was that NATO's focus was biased toward managing the capabilities (white/blue) in a way that was divisive to the ever-changing demand versus the mission driver (red/black) relationship. Rotating through the *color quadrants*, we probed for structural (what, how) and influential (who, why) aspects. This generated a wall full of sticky notes to jump start the visual modeling.

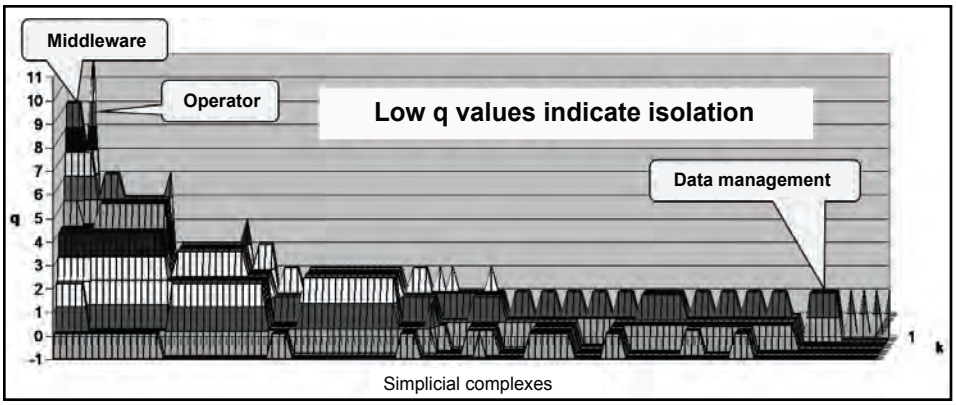
The visual models were constructed interactively in the workshops using Microsoft Visio with a custom stencil of symbols and rules applied to assure that only the allowed symbols and appropriate connectors were used within and between layers of the model. This consolidated visual model representation contained five interlocking layers:

1. **Structure/Function.** The physical structure and functioning of resources and services.
2. **Hierarchy.** The formal hierarchies and standards under which both the non-digital and digital aspects of the whole are held accountable.
3. **Trace.** The digital processes and software that interact with the physical processes.
4. **Demand.** The organization of customers' needs as demands on the way the enterprise is organized.
5. **Synchronization.** The lateral relations of synchronization and coordination within the enterprise and between the enterprise and its customers.

In between client sessions, these visual model entity-relationship diagrams were analyzed for patterns (complex or emergent[3]) that facilitate the structuring of the entities according to the stratification (from services to mission environments). The patterns revealed aligning structures between the mechanisms that determined the organization's ability to react and manage itself (e.g., governance, actors, design authority—the *determining* structures) and those mechanisms that carry out the directives of the determining structures (e.g., systems, processes, agents—the *determined* structures).

> *"The patterns revealed aligning structures between the mechanisms that determined the organization's ability to react and manage itself."*

Figure 5: *Performance Risk Landscape*



Model Matrices

The stage-one, entity-relationship-style diagrams rapidly became *eye charts*, too complex to analyze directly. The views conveyed the global complexity of the situation, providing a structural snapshot of the dynamic characteristics of this complex SoS. However, they did little to indicate specific interoperability risks among those characteristics.

In the probe's second stage, we took advantage of the defined semantics and rule-based approach of the diagramming technique to convert the diagrams into a stratified matrix. The conversion was done using an automated utility that leveraged recognizable patterns in the entity/connector relationships (e.g., a hierarchical unit that controls a synchronization of activities produced a derived entity called an orchestration). The conversion layered the connection information embedded in the diagram's semantics from the point of view of the different demands being placed on the systems; see [4] for details of this procedure.

The NATO six-level stratification is illustrated in Figure 1. The core levels—the sub-matrices labeled 1-6 (the darker shaded boxes)—form a value stratification that progresses from low-level services, systems, and know-how to high-level mission environment descriptions (this progression is represented by the connecting arrows in Figure 1). The other numbered matrices model the aligning structures that facilitate the overall enterprise's ability to react to the mission environment. The major axes are composed of the simple and complex objects[4] that model the enterprise's assets, capabilities, and processes.

Figure 1 also includes some exemplars of the entity types that populated the various sections of the matrix, such as mission situations, drivers, demand situations, constituent capabilities, events, processes (such as change notification), know-how, and outcomes.

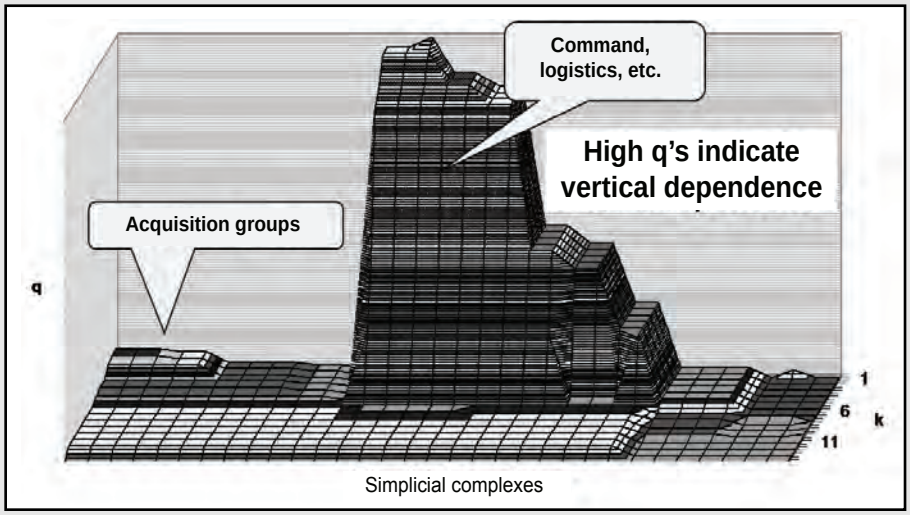
Interoperability Landscapes

In the third stage of the probe, we analyzed the stratified matrix and produced interoperability landscapes. The landscapes depict the *connectedness* of the entities, sorted so that neighboring entities show commonalities and differences in their degrees of connectedness. An interoperability landscape (like the one in Figure 2) enabled us to visualize relationships and gaps within the visual model representations, viewed from different

perspectives codified by the matrix. The columns in the landscape (Ron Atkins'[5] simplicial complexes) are organized so that entities connected through a shared number of interoperating activities are next to each other in terms of their height and depth dimensions. The height dimension (*q* in the landscape) describes the number of shared underlying activities; the higher the *q* between columns, the more related they are. The depth dimension (*k*) describes the number of other related columns there are at that level of *q*. For example, the landscape in Figure 2 shows peaks separated by valleys. These valleys illustrate the gaps between the different levels of shared activity. From this landscape and its underlying matrices, we gained insight into which 17 relationships generate the high peaks and which 10 events share services in the broad plateau on the left of the figure.

Indirect Management Control

The slicing process can be customized and creatively applied. For example, when the first-line management structures were suppressed, the impact of indirect management control jumped out. It also revealed the significant separation between the acquisition organization and the line command structures. This figure shows the vertical command dependencies in this SoS.



Outcomes About NATO Interoperability Risks from the Approach

Using the modeling and analysis techniques approach described in this article, we constructed models of the people, processes, and technologies that made up the NATO modernization program and represented the way demands were placed on their use. Using those models and representations, we developed an objective view that reflected the major interoperability challenges faced by the program, using interoperability landscapes to discover and illustrate those challenges. We categorized those challenges as Type III Mission Risks, Type II Composition Risks, and Type I Performance Risks.

Type III Mission Risks

Projecting[6] from Level 6 (mission environments) in the matrix, we examined how the SoS interoperates within its demand-driven, operational context-of-use.

Figure 3 is the three-dimensional depiction of our mission awareness landscape. This particular example shows that the predominant mission awareness integration point was the system operator and the operator's display console. The rest of the social and technical systems for areas such as development, support, and acquisition were virtually unaware of mission-demand complexity. That is, these systems did not interoperate in response to demand situations; for those that should have, the Type III risk was high[7].

Type II Composition Risks

Entering the matrix at Level 4, the orchestration level, we examined whether the systems interoperated in the ways being demanded of them.

After ordering and ranking, the resulting orchestration landscape (see Figure 4) revealed obvious islands of high connectivity with broad regions of separation. The specific entity groupings were examined to determine if the separations were warranted. For example, gaps revealed that hardware configuration management was quite separate and poorly *orchestrated* with software version management. The depth of the valleys indicates that the baseline *connective tissue* (of aspects such as change management and revision control) was far from seamless in this SoS.

The model (at the modeled fidelity) is good at indicating missing connections; it conversely indicates the presence of connections (peaks) but does not speak to the sufficiency of those connections. Therefore, gaps tend to be truer signs of interoperability risks (because it is hard to interoperate when one has no connection) than peaks are guarantees of interoperability (because high connectivity does not necessarily mean interoperability). However, both gaps and peaks are good indicators of worthy areas for further investigation.

Type I Performance Risks

Entering the matrices at Level 3, the operational capability level, we examined how the subsystems within system elements were or were not connected.

The performance risk landscape (shown in Figure 5) revealed the degree of isolation between the many structural entities in this SoS. Once again, we found a high likelihood of connectivity gap-driven risks; these gaps required further examination to determine the severity of consequences before declaring specific risk significance.

In our three categories, we identified interoperability risks and visually reinforced their presence by the landscape topologies. The objects and relationships depicted in the landscapes were familiar to the client and served to:

- Facilitate constructive dialogue about mitigation strategy.
- Justify and prioritize follow-on activities, such as detailed impact analysis, model refinement and validation (through detailed, bottom-up fact finding), and cost analysis in targeted areas.

Conclusions About the Modeling and Analysis Techniques

The examination of interoperability is a challenge in understanding complexity. The structural models produced by the techniques bring a welcome engineering rigor to the process.

In part, the effectiveness of this set of model-based analysis techniques can be attributed to the way they stress the need to speak in the client's language. The technique starts with client artifacts and builds

visual representations (entity-relationship-style diagrams) that are understood by the client. While they quickly become eye charts that are too complex to convey anything other than the global impression of complexity, these diagrams do employ rules of object-connector relationships that facilitate a transformation of the data into a stratified matrix, supporting empirical analysis. Overall, the techniques produce a rapid (nominally two days per model) snapshot of interoperability risks from the perspective of the interviewed stakeholders.

Of great merit in the techniques is the attention paid to understanding the relationship of the operational context and the supplied technologies, capabilities, and governance mechanisms[8]. By identifying gaps in their alignment, the NATO interoperability probe team identified critical risks that are often overlooked.◆

References

1. National Defense Industrial Association Systems Engineering Division, Modeling and Simulation Committee. Study Task Report: M&S Support to the New DoD Acquisition. Feb. 2004 <www.ndia.org/Template.cfm?Section=Systems_Engineering&Template=/ContentManagement/ContentDisplay.cfm&ContentID=18421&FusePreview=True>.
2. Boxer, P., and C. Eigen. Taking Power to the Edge of the Organization: Role as Praxis. Proc. of the International Society for the Psychoanalytic Study of Organizations Symposium. 2005 <www.ispso.org/Symposia/Baltimore/2005%20papers/2005Boxertext.pdf>.
3. TLCM Workstrand. Capability Management Handbook (Interim Edition), Version 2.0. United Kingdom Ministry of Defense. 26 Feb. 2007.
4. Anderson W., P. Boxer, and L. Brownsword. An Examination of a Structural Modeling Risk Probe Technique (CMU/SEI-2006-SR-017). Oct. 2006 <www.sei.cmu.edu/publications/documents/06.reports/pdf/06sr017.pdf>.
5. Anderson, W., Brownsword, L., Fisher, D., and S. Garcia. Transitionability of Complex Modeling and Simulation Approaches to Software-Intensive Systems Development (CMU/SEI-2006-SR-018).
6. Boxer, et al. PAN Projective Analysis <www.brl.com/toolsets/pan.html>.
7. Casti, John L. Complexification: Explaining a Paradoxical World Through the Science of Surprise. New York: HarperCollins, 1994.
8. International Council on Systems Engineering. INCOSE Systems Engineering Handbook: A "What To" Guide for All SE Practitioners (INCOSE-TP-2003-016-02, Vers. 2a). June 2004 <www.incose.org/ProductsPubs/products/sehandbook.asps>.

Notes

1. The techniques used were a demonstration of Boxer Research Limited's (BRL's) Projective Analysis (PAN). The SEI and BRL have since integrated uses of PAN into the SEI SoS Navigator suite of products. PAN has been applied in enterprises in such domains as manufacturing, health care, defense, and telecommunications [2]. For example, PAN has been used in support of Through Life Capability Management practices [3] for the United Kingdom Ministry of Defense. Projects under the European EUREKA program, jointly funded with the Department of Trade and Industry in the UK, with City University (London) as a subcontractor, further developed parts of the technology.
2. This article has been drawn, with permission, from two SEI special reports [4, 5] and from information about PAN available at [6]. We have used figures utilized in our NATO research. To protect NATO's confidentiality, some actual specifics have been removed: For example, the vertical marks above simplical complexes in the landscape figures each represent an item with a name that has been removed.
3. The patterns are represented by model-generated entities emerging from more complex interactions than are represented by simple entity-connector-entity constructs (e.g., markets, orchestrations, and super-channels).
4. Tangible assets, such as control modules or design expertise, are named *simple objects*. Patterns of relationships that form outcomes or mission situations are named as *complex objects*.
5. The form of description behind this matrix format uses the mathematics of Ron Atkins' Q-analysis. An introduction to this can be found in [7]. The simplicial complexes are derived directly from the named entities in the visual model and from the patterns of objects generated by the stratification process.
6. *Projection* is the systematic process used to examine the matrix representation of the modeled SoS; it is described in detail in [4]. The technique uses the stratification to provide *entry points* at different levels of complexity. This provides a means by which the interoperability issues can be decomposed at different levels of complexity.
7. If the consequence of the detected condition is not serious (i.e., benign), the risk may be considered low [8].
8. The technique models the structure-determining processes of the organization-in-context as well as the structure-determined processes of the systems the organization uses.

About the Authors

William B. Anderson has a career-long focus on process improvement and technology management. Anderson is a senior member of the SEI technical staff, where his research interests include the integration and interoperability of complex software-intensive systems, service-oriented architecture and reuse management, and the modeling of complex systems.

SEI
Carnegie Mellon University
4500 Fifth AVE
Pittsburgh, PA 15213-3890
Phone: (412) 268-5386
Fax: (412) 286-5758
E-mail: wba@sei.cmu.edu

Philip Boxer, a Certified Management Consultant and senior member of the SEI technical staff, has advised on strategy since 1980, supporting leadership teams across both public and private industry sectors in bringing about transformational change. His focus is on the challenges organizations face from asymmetric forms of demand, and on the mitigation of risks associated with failing to develop requisite agility.

SEI
Carnegie Mellon University
4500 Fifth AVE
Pittsburgh, PA 15213-3890
Phone: (412) 268-5386
Fax: (412) 286-5758
E-mail: pboxer@sei.cmu.edu

Quality and Cost – It's Not Either/Or: Making the Case With Cost of Quality

George Webb
45th Range Management Squadron, Patrick Air Force Base

LTC Nanette Patton
Office of the Surgeon General

Today's organizations must be committed to the continuous pursuit of quality improvement as a requirement for survival. Traditionally, quality and cost have been perceived as a trade-off decision. For this reason, the main purpose and benefit of measuring quality costs has been to demonstrate that improved quality and lower costs go hand-in-hand. Through collection and analysis of these quality costs, improvement is translated into a language management listens to and responds to: money. This article provides tools and techniques to help infuse cost of quality (COQ) concepts into the project team activities to promote quality improvement throughout the full project life cycle.

There have been many changes in how DoD project management applies quality to software projects. In days gone by, there was a separate cost for quality attached to the items identified within an acquisition, whether for software or for hardware. As acquisition, project management, and system engineering have evolved, many companies have replaced the term *quality* with other terms, such as *performance* and *best practices*. Current best practices standard-bearers—such as the International Organization for Standardization (ISO), the Software Engineering Institute's Capability Maturity Model® Integration (CMMI®), the Project Management Institute (PMI), and the IEEE—integrate quality into everyone's work ethic and processes. This entails creating integrated product/process teams (IPTs) and letting the individuals, such as engineers, logisticians, configuration managers, testers, and project mangers (PMs), assume responsibility for quality within their functional processes.

ISO and CMMI argue for a separate quality group to maintain objectivity. While the PM has the final responsibility for quality, a quality manager can be responsible for day-to-day quality by developing and implementing a quality effort. Experience has shown, however, that when funds are tight and time is short, the quality group is among the first cut because they do not produce a *physical product* for the customer. Another short-term cost-saving measure is for the PM to select someone untrained in quality assurance, from the ranks of the engineering staff, to be the quality manager. In other cases, there may be no identification at all of a separate quality process owner to oversee this critical area; consequently, the COQ remains hidden in other project costs. When quality is just another subprocess, it has to fight for attention, priority, and funding like all the others. When this occurs, the opportunity to identify and correct problems early in the life cycle is often lost.

This maladaptive application of quality principles can be attributed to a lack of training, management support for quality processes, and adequate quality cost measurement systems. It is more likely to occur when cost and schedule become more important than or equal to quality.

Figure 1 (taken from [1]) shows the basic cost of good and poor quality.

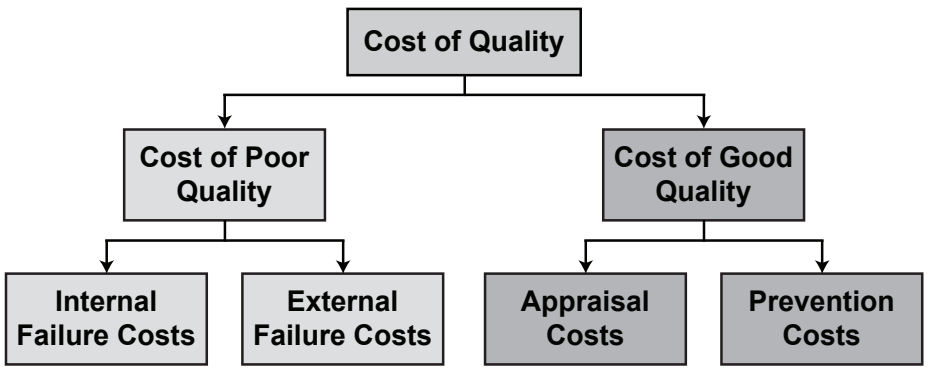
Evolution in the Approach to Quality

In decades past, the focus of quality was merely finding problems at the end of an assembly line and removing the defects before shipping to the consumer. If the product did not meet specifications, it was either reworked or scrapped—both expensive options. This approach is prone to human error and rarely finds all defects. Furthermore, this quality control approach only identified the defects found through a random sampling, but actually did nothing to determine the root cause of the problem for resolution.

If preventive quality measures and rework are deferred until the testing phase, the cost of change is 40 to 100 times greater than if the defect was fixed when it was created [2]. The testing stage has the least recovery time for *show-stopper* problems or unexpectedly large amounts of rework. This unpredictability becomes a large contributing factor to why projects miss their schedules. Furthermore, this type of approach, which assumes that doing more testing leads to shipping a better product, only works up to a point. With pure testing, one can get to something approaching 5-Sigma quality (0.2 defects per thousand lines of code); however, a product shipped at 5-Sigma is perceived as inadequate by today's manufacturing standards [3].

In the post-World War II reconstruction years, Dr. W. Edwards Deming introduced a quality program that simultaneously controlled the production and quality processes. Unfortunately, the United States did not adopt these principles until the 1980s with the introduction of the Total Quality Management System. Deming's core message—that we should stop inspecting defects out of products and start building quality in—has remained. The common thread of various quality methodologies is that the project team will build quality into the system design and will address quality continually throughout

Figure 1: *Cost of Quality*



® The Capability Maturity Model and CMMI are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

the life cycle. The goal is to identify problems up front and early, allowing corrective action and quality prevention to take place to reduce the number of critical defects found at the end of the assembly line.

This goal can be met through software quality surveillance, which includes walkthroughs, peer reviews, inspections, testing, IPT structures, as well as any method that identifies quality problems, risks, and operational capability weaknesses as early as possible. Approaching quality in this manner provides early corrective action and promotes lower quality costs upfront and early, thereby reducing end-of-program cost overruns [4].

Today, we have gone a step further by identifying risks which may have the potential to change engineering requirements, operational capabilities, and the quality of the product. In the spirit of risk management, software developers can help prevent one of the most common causes of defects—ambiguous requirements—by writing comprehensive acceptance tests when recording each requirement. Furthermore, automating these tests and running them as part of frequent integration builds will help detect defects when they happen.

While common sense says that preventing defects or finding them when they are cheapest to fix is preferable to finding them at the end when they are many times more expensive, several software development projects fail to write tests upfront, do inspections, or perform frequent integration—despite the benefits.

Why We Don't Implement Preventive Quality Processes

Implementing quality processes is tedious, time-consuming work in most environments. And, time is money. There is document inspection (usually several hundred pages) and writing early tests for critical requirements at the beginning of a project. It is hard to keep the tests up-to-date as the requirements change and even harder when you realize that you have to inspect the tests. These strategies increase the cost of implementing quality and the return on investment is not always predictable with a high degree of reliability, especially when the requirements and design have not been locked in. Thus, it is mind-wrenching work to determine which of all the possible strategies for implementation will bring the best value to the project.

Carolyn Fairbank, CEO of the Quality Assurance Institute, said:

> We're far too focused on product delivery, not process capability. We're too busy trying to get the product out the door. Granted, this is a market-driven phenomenon, but we'll have to change that deadline-driven attitude to one of good processes. If you get the process right, the product will have a far better chance at success. Unfortunately, many IT professionals still don't quite understand the concept of process management. [5]

According to Karl Wiegers:

> We do far too much pretending in software. We pretend we know who our users are, we know what their needs are, that we won't have staff turnover problems, that we can solve all technical problems that arise, that our estimates are achievable, and that nothing unexpected will happen. Risk management is about discarding the rose-colored glasses and confronting the very real potential of undesirable events conspiring to throw our project off track. [6]

"Today, we have gone a step further by identifying risks which may have the potential to change engineering requirements, operational capabilities, and the quality of the product."

Risk identification requires a look into the future as to the potential success of the program. The challenge lies in the identification of risk versus current problems. The PMI defines risk as "an uncertain event or condition that, if it occurs, has a positive or negative effect on a project's objectives" [7]. Current problems require attention and action even if the immediate remedy is to defer corrective action until later. Risks realized may require actions that lead a project team to proceed in a different direction altogether or canceling the project entirely. Project teams must accept some risks due to other requirements, conditions, assumptions, or constraints; however, if a project team chooses to completely ignore risk, they greatly increase the probability of project failure.

A company's economic status or condition are significant factors when deciding what process to implement to track quality cost [8]. Pursglove and Dale suggest that the profitable nature of the business can make it more difficult to convince management of the need to track COQ [9]. For example, having more engineers and fewer quality assurance people on a project can be great for a company's short-term financial success. However, if project staff members do not build in quality from the start, a greater reliance on product rework results. The organization will eventually pay for the inadequate quality as customers identify problems with the product or service. Engineering changes must take place before the customer deems the product usable. These engineering changes late in the development process may result in a product or service that does not quite meet the original intent on the capabilities delivery; this, in turn, can lead to lost business. If this happens on a recurring basis, the company may experience competitive and financial difficulties. If so, a company may be more open to performing an assessment in an attempt to get back on track. Then, after collecting and analyzing data that reveals a quality problem, the company finally decides to track quality costs. This may also be the time when the company experiences total failure. They know something has to be done, but don't have a well thought-out plan. They make *knee-jerk* decisions, such as simply canceling the project and not addressing the underlying quality problems in their processes; that, in turn, causes unintended conflict within the organization.

Not having a clear understanding of the actual value of COQ also hinders the adoption of quality processes. There has been a persistent misconception in the business community that the COQ is a cost over and above that of developing and producing a project to meet a specific and required outcome and schedule. The COQ, regardless if it is software or hardware, is the price of not creating a quality product or service. If the development process was perfect with no problems and there was no possibility of substandard service, failure of products, or defects in their manufacture, then organizations would have no expenditures on COQ.

Prevention	Appraisal
Represents everything a company spends to prevent software errors, documentation errors, and other product-related errors. • Staff training • Requirements analysis • Early prototyping • Fault-tolerant design • Defensive programming • Usability analysis • Clear specifications • Accurate internal documentation • Pre-purchase evaluation of the reliability of development tools	Includes the money spent on the actual testing activity. Any and all activities associated with searching for errors in the software (and associated product materials) fall into this category. • Design reviews • Code inspection • Glass box testing • Black box testing • Beta testing • Test automation • Usability testing • Pre-release out-of-box testing by customer service staff
Internal Failure	**External Failure**
The cost of coping with errors discovered during development and testing. These are bugs found before the product is released. • Bug fixes • Regression testing • Wasted in-house user time • Wasted tester time • Wasted writer time • Wasted marketer time • Wasted advertisements • Direct cost of late shipment • Opportunity cost of late shipment	The costs of coping with errors discovered after the product is released. These are typically errors found by your customers. • Technical support calls • Answer books (for support) • Investigating complaints • Refunds and recalls • Interim bug fix releases • Shipping product updates • Warranty, liability costs • Public relations to soften bad reviews • Lost sales • Lost customer goodwill • Supporting multiple versions in the field • Reseller discounts to keep them selling the product

Table 1: *COQ Data Points*

COQ is the sum of costs incurred in maintaining acceptable quality levels plus the cost of failure to maintain that level (cost of poor quality), and typically ranges from 15-25 percent of total cost [10].

Philip B. Crosby's "Quality Is Free" concept [11], identified two main categories of quality costs: conformance costs (cost of good quality), and nonconformance costs (cost of poor quality).

Conformance costs include prevention and appraisal costs; nonconformance costs include *internal failures* as well as *external failures* (Table 1, from [12]). A defect found early in the project prior to customer delivery is termed an internal failure. A defect identified after the product has been deployed to the customer is an external failure. External failures can also include incompatibility of the software with legacy software installed in the field, or a lack of commonality between redundant systems.

Beyond not clearly understanding COQ concepts, key decision makers in an organization may lack knowledge in determining quality costs and the principles for collecting quality costs. Without knowing what quality principles are, an individual or organization may have no idea where to place their focus to obtain quality costs. The organization can remedy this either by ensuring that a quality curriculum is included in the training for project staff and senior leadership or by hiring a quality consultant to guide the organization.

Getting a COQ System in Place

Herb Krasner and Dan Houston explain that companies need to answer three questions [13]:

1. How much does poor software quality cost?
2. How much does good software quality cost?
3. How good is our software quality?

Once these questions are answered, the project team can compare quality costs to overall software production costs and software profits, and to benchmarks and norms. They can also better analyze product quality to improve their competitive situation, measure improvement actions and the bottom-line effect of quality programs, visibly see previously hidden costs related to poor quality, and more clearly see the economic tradeoffs involved with software quality.

Even if the project team cannot measure all of these costs with a high reliance, a COQ model quantifies (for management and executives) the amount of money being lost on fixing defects and delivering poor-quality products. This, in turn, negatively affects their bottom line. That provides motivation and impetus for implementing preventive quality measures.

In order to feed decision makers those costs with some degree of legitimacy, a life-cycle model has been developed to guide this work (Figure 2, next page).

Note: If no data source exists for the collection of costs, then the project team will have to use some type of analytical technique to develop a cost estimate model. With this model, the team should be able to establish the cost estimates for the appropriate quality categories.

Capturing COQ

This step in the life cycle is twofold: (1) identify the costs and (2) determine a method for entering costs into an accounting system that tracks them throughout system/service development.

When quality costs are initially determined, the categories included are the visible ones.

Oftentimes it is what you do not know that can hurt a project. Software quality costs are not always easy to identify within programs. Software has many hidden costs that may not be readily apparent to the project manager. These are shown below the water line in Figure 3 (next page, adapted from [1] and [14]) and in the expanded Figure 4 list on page 27.

As an organization internalizes a broader definition of poor quality, the hidden portion of the iceberg becomes apparent. Identifying these costs opens a

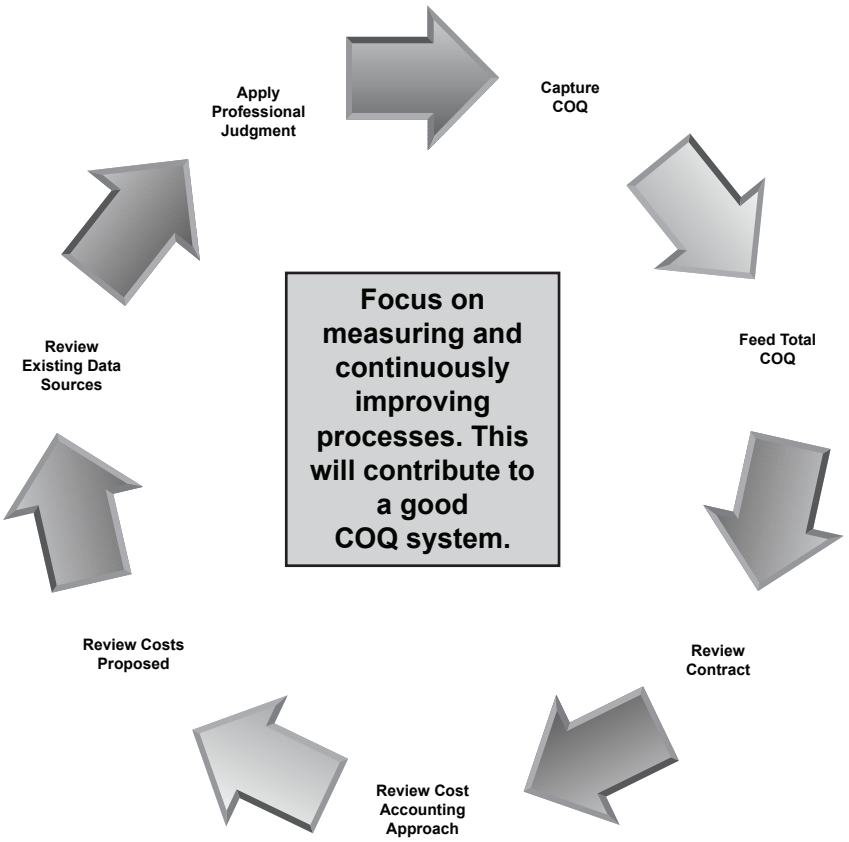


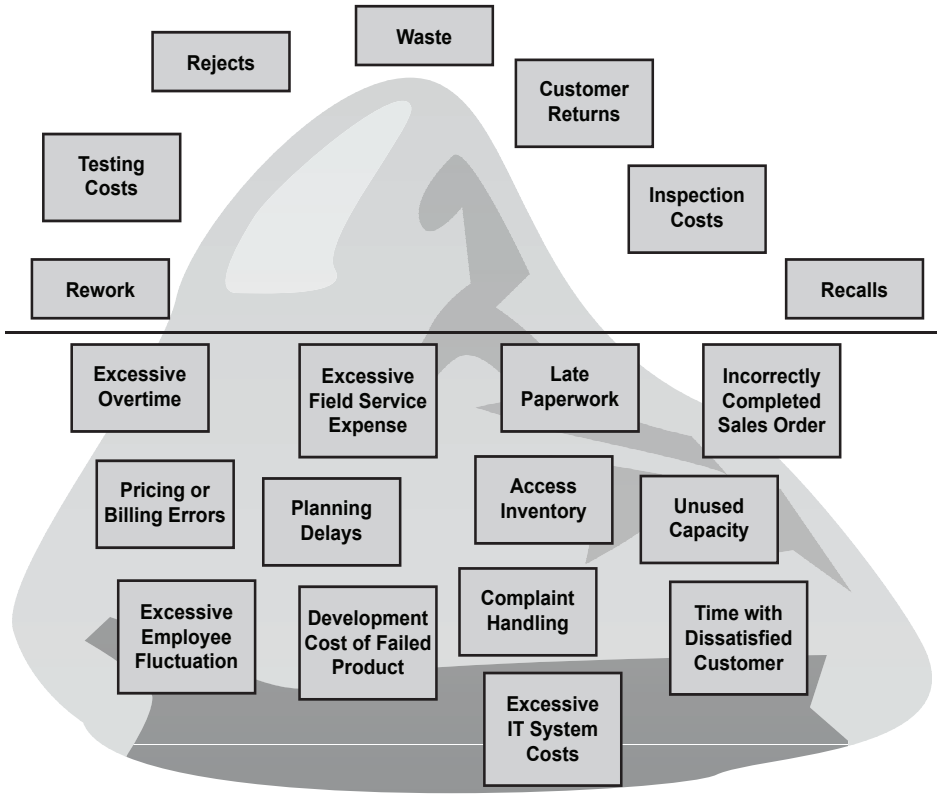
Figure 2: *The Life Cycle of COQ*

door of opportunity and the project team can then implement processes to avoid more costly expenditures later in the project. Furthermore, the team increases the probability that the product/service will be acceptable to the customer within all required elements and functions that the product/service is supposed to deliver.

A proper focus on quality entails identifying and funding quality costs and institutionalizing quality processes at the very beginning. It also requires that quality experts supply to management an estimate of the total quality costs, good or bad. Management uses the information obtained through the quality initiative as a tool to adjust funding allocations. Once the quality experts have gathered data, upper management can determine with a clearer picture where to concentrate quality efforts and funding for eventually achieving a greater positive impact and promoting a successful project.

While the *traditional method* is the most used method in collecting costs related to quality (according to the American Society for Quality), organizations can also use the defect document collection method, the time and attendance collection method, or the assessment method. The PM and the quality manager (if the PM has designated one for the project), will have to do a cost-benefit analysis to determine which method is the best fit for the organization and project, given available resources.

Figure 3: *The Iceberg Model of COQ*



Feed Total COQ

Having identified costs and a method for tracking them, diligent data collection is now required. This also includes incorporating previously unidentified costs revealed during ongoing activities that now require tracking.

Review the Contract

The quality group must be conscious of the legal terms as well as the performance of specific tasks within any contract to support product/service development and delivery. They should be familiar with specifics to ensure the contractor is completing required tasks throughout the life cycle of the product. There may be specific tasks that occur sporadically during the development cycle and therefore require a more concerted follow-up.

Review the Cost Accounting Approach

On a periodic basis, it is important to ensure that the cost accounting process and repository adequately track all pertinent cost information collected on work currently performed.

The system may require refinement due to new requirements and/or additional costs not identified at the start of the life cycle. This may also require changes in the data collection method.

Review Existing Data Sources

It is also important to ensure that the sources used for cost data provide the best

available data in terms of validity and accuracy. Do not hesitate to switch to a better data source if it provides data that will give a more accurate picture of where the project stands at a particular point within the process.

Review Costs Proposed

As development continues, hidden costs not identified at the start will reveal themselves. Track these costs along with those originally proposed to facilitate budget adjustments and to recalculate the return on investment projections in order more effectively manage expectations.

Studies (such as [9]) indicate that the further along in the process quality is worked into the product or service that the higher the COQ will be. The project team should try to reduce the overall cost of each product or service by establishing the optimum level of preventive and appraisal costs that minimizes resultant error costs. The net result of quality improvement should be a reallocation of costs across the COQ categories resulting in a reduction in the overall COQ. An example of this [15] is shown in Figure 5.

Apply Professional Judgment

This is the time for analysis of all information gathered regarding the health of the program. By pushing this data and analysis to the appropriate project decision makers, they can make informed decisions on how to proceed to ensure project success.

Conclusion

At first glance, an individual might be prone to think that collecting quality costs is expensive, adding unnecessary costs to the product or project. Quality is not free in that you have to make an up-front investment in time, money, and effort. However, if performed properly over the full life cycle of the project, you can recoup the resources expended for quality processes by avoiding rework later in the project life cycle. By communicating the quality story in terms of dollars, you can enlist the help of senior management to infuse quality processes throughout the project life cycle and contain project costs for the long haul. As with many project management standards and guides (e.g., [7]), collecting quality costs is like project planning; it is cheaper to properly plan than it is to plan a little and fail a lot.◆

- Excessive overtime
- Pricing or billing errors
- Premium freight costs
- Excessive field service expenses
- Planning delays
- Low morale
- Excessive employee turnover
- Development cost of failed product
- Overdue receivables
- Excessive system costs
- Complaint handling
- Expediting costs
- Paperwork/documentation is late
- Customer allowances
- Lack of follow-up on current programs (require re-testing)
- Excess inventory
- Unused capacity/scrap cost
- Time with dissatisfied customers
- Incorrectly completed sales order
- Scheduling delays
- Test tool certifications
- Training of operators
- Obtaining user licenses for off-the-shelf software
- Security measures for information assurance
- Designated Approving Authority certifications
- Compatibility issues with legacy
- Government oversight (for government contracts)
- Risk tracking and impacts of external dependencies
- Management reserve for tracking risks
- Training in risk detection
- Possibility for developing more quality gates in development, which will require additional personnel and time.
- Bug fixes
- Regression testing
- Wasted in-house user time
- Wasted tester time
- Wasted writer time
- Wasted advertisements
- Direct cost of late shipment
- Opportunity cost of late shipment

Figure 4: *More Hidden Quality Cost Data Sets Not Easily Identifiable*

Figure 5: *COQ in Software Testing*

	A	B	C	D
1	Testing Investment Options: ROI Analysis			
2				
3		No Formal	Manual	Automated
4	Testing	Testing	Testing	Testing
5	Staff	$0	$60,000	$60,000
6	Infrastructure	$0	$10,000	$10,000
7	Tools	$0	$0	$12,500
8	Total Investment	$0	$70,000	$82,500
9				
10	Development			
11	Must-Fix Bugs Found	250	250	250
12	Fix Cost (Internal Failure)	$2,500	$2,500	$2,500
13				
14	Testing			
15	Must-Fix Bugs Found	0	350	500
16	Fix Cost (Internal Failure)	$0	$35,000	$50,000
17				
18	Customer Support			
19	Must-Fix Bugs Found	750	400	250
20	Fix Cost (Internal Failure)	$750,000	$400,000	$250,000
21				
22	Cost of Quality			
23	Conformance	$0	$70,000	$82,500
24	Non-conformance	$752,500	$437,500	$302,500
25	Total CoQ	$752,500	$507,500	$385,000
26				
27	Return on Investment	#N/A	350%	445%

References

1. Buthmann, Arne. iSixSigma. "Cost of Quality: Not Only Failure Costs." <http://europe.isixsigma.com/library/content /c070502a.asp>.
2. Boehm, Barry W., and Richard Turner. Balancing Agility and Discipline: A Guide for the Perplexed. Addison-Wesley Professional, 2003.
3. Lahman, H.S. "Testing vs. Defect Prevention." 27 Oct. 2004 <http://pathfinderpeople.blogs.com/hslahman/testing_and_defect_prevention/index.html>.
4. The Data and Analysis Center Gold Practices Web site. "Software Acqui-

Coming Events

January 10-13

6th Annual IEEE Consumer Communications and Networking Conference
Las Vegas, NV
www.ieee-ccnc.org

January 19-20

16th Annual Multimedia Computing and Networking Conference
San Jose, CA
http://mirage.cs.uoregon.edu/mmcn2009

January 19-22

10th Annual Lean Six Sigma and Process Improvement Summit
Orlando, FL
www.site-members.com/lsspi/index.html

January 21-23

Principles of Programming Languages
Savannah, GA
www.cs.ucsd.edu/popl/09/

January 27-30

Network Centric Warfare 2009
Washington, D.C.
www.ncwevent.com/index.php

April 20-23

21st Annual Systems and Software Technology Conference
Salt Lake City, UT
www.sstc-online.org

Coming Events: **Please submit coming events that are of interest to our readers at least 90 days before registration. E-mail announcements to: nicole.kentta@hill.af.mil.**

sition Gold Practice, Formal Risk Management." 2008 <https://www.goldpractices.com/practices/frm>.
5. Smith, Kennedy. "The Software Industry's Bug Problem." Quality Digest. Apr. 2003 <www.qualitydigest.com/april03/articles/03_article.shtml>.
6. Wiegers, Karl. "Know Your Enemy: Software Risk Management." Software Development. Vol. 6, No. 10. Oct. 1998: 38-42 <www.dci.pomona.edu/~markk/cs121.f07/supp/risk_mgt.html>.
7. Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK Guide). 3rd ed. PMBOK Books, 2004.
8. Campanella, Jack. "Principles of Quality Costs: Principles, Implementation, and Use, Third Edition." ASQ Quality Cost Committee. ASQ Corporation, 1990.
9. Pursglove, A.B., and B.G. Dale. "The Influence of Management Information and Quality Management Systems on the Development of Quality Costing." Total Quality Management Vol. 7, No. 4, 1 Aug. 1996: 421-432.
10. DeFeo, Joseph A. "The Tip of the Iceberg." Quality Progress Vol. 34, No. 5, May 2001: 29-37.
11. Crosby, Philip B. "Quality Is Free: The Art of Making Quality Certain: How to Manage Quality So That It Becomes A Source of Profit for Your Business." Mentor. 1 Jan. 1980.
12. Pirozzi, Rob. "Understanding Quality Cost." LogiGear Corporation. 23 Oct. 2007 <www.logigear.com/newsletter/understanding_quality_cost.asp>.
13. Krasner, Herb, and Dan Houston. "Using the Cost of Quality Approach for Software." CrossTalk Nov. 1987.
14. DeFeo, Joseph A. "The Tip of the Iceberg." Quality Progress Vol. 34, No. 5, May 2001: 29-37.
15. Black, Rex. "Investing in Software Testing: The Cost of Software Quality." 2000 <www.compaid.com/caiinternet/ezine/cost_of_quality_1.pdf>.

About the Authors

LTC Nanette Patton is currently serving as the Army Medical Department Deputy Chief Information Officer for Army Systems Integration in the Office of the Surgeon General (OTSG). LTC Patton has over 10 years experience as a health services systems manager and is Level III DoD-Acquisition-certified in Information Technology and Level II-certified in Program Management. She has master's degrees in human resources management from Chapman University and in business administration (with a concentration in computer information systems) from Colorado State University.

OTSG
Information Management Directorate
5109 Leesburg Pike
STE 594
Falls Church, VA 22041-3258
Phone: (703) 681-0187
E-mail: nanette.patton@us.army.mil

George Webb is currently serving as lead software quality assurance specialist for the 45th Space Wing Range Management Squadron, Patrick Air Force Base, Florida. Webb is Level III Defense Acquisition Workforce Improvement Act-certified in Production, Quality, and Manufacturing and Level I-certified in Logistics and Program Management. He also has a Business Management Certification for the State of Florida. Webb has worked as a quality assurance specialist for the DoD for 22 years and has focused his efforts on software quality for the past decade.

45th Space Wing Range Management Squadron
14640 Hangar RD
Canaveral Air Station, FL 32925-2206
Phone: (321) 494-0755
E-mail: george.webb@patrick.af.mil

Letters to the Editor

Dear CrossTalk Editor,

The April 2008 article by Kym Henderson and Dr. Ofer Zwikael—*Does Project Performance Stability Exist? A Re-examination of CPI and Evaluation of SPI(t)* [Schedule Performance Index (time)] *Stability*—is based on faulty research. While serving as a senior program analyst for contract performance management at the Office of the Secretary of Defense, I sponsored research on Earned Value Management (EVM) analysis (by Dr. David Christensen and his Air Force Institute of Technology students) and provided access to the defense contract database.

Henderson and Zwikael cite "an internal DoD [Naval Air Systems Command] research project" by Michael Popp and state, "In contrast to Christensen and associates research, which used data from the DAES (Defense Acquisition Executive System) database, the data used by Popp was sourced from the Contracts Analysis System database ... " They are mistaken: DAES and CAS are the same database under different names, now part of the Defense Acquisition Management Information Retrieval system.

Building on earlier work, Christensen and Captain Scott Heise published their benchmark paper "Cost Performance Index (CPI) Stability" in the *National Contract Management Journal*. They analyzed data from 155 contracts that met DoD EVM requirements and distinguished between contracts having stable and unstable performance measurement baselines. Henderson and Zwikael do not explain how Popp selected contracts from the same source.

They also include information on 37 projects from Israel, the United Kingdom, and Australia. I suggest that these disparate projects did not implement EVM consistently, as on DoD contracts, and that the analysis lacks rigor (Israeli data were analyzed "using visual inspection of charts").

Henderson and Zwikael claim to have "overturned long-standing findings and beliefs on CPI stability." I disagree. Christensen and Capt. Heise have shown that CPI stability exists on DoD contracts. Whether that "rule" is true in other management environments is uncertain, but the article does not make the case for refuting it.

—Wayne F. Abba
<abbaconsulting@cox.net>

Dear CrossTalk Editor,

We would like to respond to Mr. Abba's letter.

The article's purpose was to re-examine the CPI stability rule (the rule) to see whether the Earned Schedule SPI(t) exhibited consistent stability behavior. The rule has been claimed in various authoritative sources to have unqualified universal applicability for, as we stated in the article, "all projects using the EVM [Earned Value Management] method."

The article's conclusion that the referenced claims of unqualified universal applicability of the rule cannot be generalized as self-evident. Notably, Abba's statement of uncertainty on whether the rule "is true in other management environments" significantly revises the previous unqualified claims of its universal applicability.

Abba's letter illustrates that the rule has been observed by one principal researcher using data (with one exception) from the DoD DAES database to which access for research is restricted. Christensen's research has, to the best of our knowledge, not been independently corroborated by other researchers, an important prerequisite for claiming general applicability.

Mr. Popp's Naval Air Systems Command research provided a unique independent view of the applicability of the rule utilizing data now known, thanks to Abba's correction, to also come from the same restricted data source utilized by Christensen.

Popp's report corroborates the research using commercial sector EVM data and practitioner observations that the CPI stability rule does not always apply on their projects. Abba's letter provides no basis for modifying the paper's conclusions.

The Popp report and Christensen research highlight many occasions for DoD projects where the rule applies. Follow-on research was recommended to ascertain any characteristics which result in early CPI stability and also where a progressively improving CPI precluded stability.

This, with Earned Schedule method research, provides opportunities for enhancing project management practice, project performance, and the application of EVM. We encourage more studies in various project environments focusing on this important—still to be fully solved—area.

—Kym Henderson
<Kym.Henderson@froggy.com>

—Dr. Ofer Zwikael
<ofer.zwikael@vuw.ac.nz>

Dear CrossTalk Editor,

In your June 2008 edition, you asked for stories on "how we have helped." Well, here goes: CrossTalk is considered our base resource for the two Software Acquisition Management courses here at the Defense Acquisition University. Almost every lesson relies on both past and present CrossTalk articles. Many lessons learned and best practice stories have been extracted from CrossTalk and used as anecdotes in our classes. Students and instructors tell us that they use it to find out the latest happenings in the DoD software industry. It is an invaluable tool that allows DoD software professionals to keep up with the fast pace of technology. The fact that software is taking over the functionality in all of our major systems makes CrossTalk extremely important to the DoD software acquisition professional!

—Bob Skertic
<Bob.Skertic@dau.mil>

Dear CrossTalk Editor,

While I agree that some improvement is better than no improvement, I don't agree with the idea—from *Quality Processes Yield Quality Products* (June 2008) by Thomas D. Neff—that any process improvement model can be used by any organization. I also don't agree with Neff's ideas to "just do it," and that "it doesn't matter" which model you choose.

Over the last 25 years, I have used, taught, and consulted on several management models, each having assumptions and values built into them. By the same token, each client organization has taken-for-granted assumptions and values built into its culture.

To achieve maximum bang for the buck, organizations are well-advised to take a few minutes to figure out who they are before they select a process improvement model. With that self-knowledge in-hand, an organization can seek out a model that best fits its culture. The closer the match, the better perceived outcomes will be.

—Gaylord Reagan
<greagan@attglobal.net>

Clouds From Both Sides, Now

I have a strange quirk, one of several. When I read sophistic technical, political, or commercial articles, I often associate them with a song stowed in my cerebral archive. More accurately, I start hearing the song crescendo in my head to the point I can't concentrate on the article. Such is the case with the latest flood of cloud computing articles.

With big guns like Google, Microsoft, Sun, IBM, Dell, and Amazon seeding cloud computing technology, the forecast is partly cloudy to overcast with an eventual downpour.

Every time I try to get my head into cloud computing, I hear Greg Lake's carnival barker voice from the Emerson, Lake & Palmer song "Karn Evil 9."

> Welcome back my friends
> To the show that never ends
> Were so glad you could attend
> Come inside, come inside! [1]

I'm sure you can help me see the light ... or cloud the light? I'm not sure. Distributed computing? Not a problem. Computing Grid? Makes sense for large scale operations. Software-as-a-Service? Emphasize service, and I'm on board. However, cloud computing and its *everything-as-a-service* mantra sounds more like mountebank bellows than a reasoned thesis.

Part the cloud jargon and I see computing rental, leasing, and sharing. You basically rent and share processing time, data storage, security, platforms, applications, and so forth. I know it's so much more than that, but is it?

You don't own it, you can't control it, and it has no intrinsic resale or depreciation value. So why not use rental, lease, or timeshare nomenclature? Not sexy enough, too transparent, or is it burdened with negative connotations of rental and timeshare property? "Yes you, Joe Shmuck, can have your own slice (for two weeks) of paradise anytime you want (except weekends, holidays, on sunny days, and during picturesque sunsets)."

What are the anticipated advantages of cloud computing? Topping the list is the reduction in IT capital expenditures. No servers, computers, applications, storage, or personnel to maintain them. Well, you will likely need some type of computer to connect to the cloud and support to maintain and configure your computers to assure cloud compatibility.

Where do those costs go? The cloud service provider (CSP) picks them up and passes them back to you in a little-extra-fee-for-service. Hey, silver linings are not free! Of course, your monthly cloud bill will be as clear as your cable, phone, or favorite utility bill.

How about device independence? Do you really think I'll be able to use my iPhone to access any cloud? "Sorry sir, iPhones only work with iClouds. For cloud nine you will need the new Nimbostratus from Micro-cloud."

What about access to supercomputing power and flexibility? You want supercomputing? "Miss, supercomputing is for Silver Cloud Members only. All *-ilities* have their own surcharge and you will be required to purchase a cloud security undercoating."

What about the efficiencies of centralization? You mean multi-tenancy? Sure six guys can pool resources to rent an apartment easier than one, but that does not guarantee optimal conditions. When everyone wants to run payroll, invoices, or crunch out the human genome at the same time—not that it would ever happen—how efficient can that be? There is a reason the American Dream involves single-tenancy, independence, and self-determination. For me, I'll stick with my own little cumulus humilis (a.k.a. a fair weather cloud) and compute along. And, if by chance, we meet on the super jet stream, remember: I'm old school—hey you, get off of my cloud!

I love the argument, "you don't generate your own electricity so why generate your own computing?" My short answer: because I can. My long answer: because electricity is a utility and I'm not ready to concede that the tools I use to create, design, produce, and improve are mere utilities. When I tinker with, tweak, and maintain my computer, I discover new ways to use it and open new doors. Would you ask a Jedi to give up his light saber to timeshare a cloud saber?

How about interoperability? Let's see, there are cloud applications, cloud services, cloud platforms, cloud storage, and competing CSPs. How will these be interoperable, or, "how do you catch a cloud and pin it down?" [2] Wait, I know! The high capital costs to start a CSP will provide a barrier to the CSP market and drive the industry to an oligarchy, monopoly or ... gasp ... a Googleopoly. Then you will have interoperability via collusion, soon followed by National Cloud Care, the Cloud Protection Agency, Cloud Footprints requiring Cloud Offsets, and No Cloud Left Behind.

You can't beat the reliability of cloud computing's multiple servers, sites, and Continuity of Operations Plans (COOP). Oh yes, the ever-present Cloud COOP. Well, if the reliability and service of present-day ISPs are any indication of the reliability and service of future CSPs, then I suggest hip waders. "Sir, I'm looking at our cloud and it is up and fully functional. The problem must be with your thin client. Unfortunately, our cloud coverage does not extend to thin, lean, or slim clients."

Ask Amazon's S3 clients (July 2008) or the London Stock Exchange (September 2008) about reliability and the effects of software as a non-service. But what's a million-dollar trade commission loss amongst cloud members? Besides, that was not real cloud computing, just contrails.

Listen, can you hear Joni Mitchell? She's a true cloud aficionado:

> I've looked at clouds from both sides now
> From up and down, and still somehow
> It's cloud illusions I recall
> I really don't know clouds at all. [3]

Don't let me rain on your parade or cloud your judgment. You may find a silver lining in them thar computer clouds. I've been wrong before and I'm sure I'll be wrong again. If you must ... send in the clouds ... there ought to be clouds ... well, maybe next year.

—Gary A. Petersen
Arrowpoint Solutions, Inc.

References

1. Emerson, Lake & Palmer. "Karn Evil 9 (1st Impression Part 2)." Brain Salad Surgery. Atlantic Records, 1973.
2. Rodgers, Richard, and Oscar Hammerstein II. "Maria." The Sound of Music. 1965.
3. Mitchell, Joni. "Both Sides, Now." Both Sides Now. Warner Brothers, 2000.

CrossTalk / 517 SMXS/MXDEA
6022 Fir AVE
BLDG 1238
Hill AFB, UT 84056-5820

PRSRT STD
U.S. POSTAGE PAID
Albuquerque, NM
Permit 737

CrossTalk is co-sponsored by the following organizations: